Fachhochschule Köln
Cologne University of Applied Sciences

Institut für Informatik

Campus Gummersbach

Studiengang: Allgemeine Informatik

vorgelegt an der Fachhochschule Köln

Bachelorarbeit

Zur Erlangung des akademischen Grades eines

Bachelor of Science

„Analyse und Vergleich von nativer Entwicklung auf dem Android-Betriebssytem mit der plattformunabhängigen Entwicklung mit dem Titanium SDK am Beispiel des Mobile SQL-Trainers“

ausgearbeitet von: Matthias Spies

Matrikelnummer: 11075052

Telefon: +49 1577 5322648

Erste Prüferin: Prof. Dr. Heide Faeskorn-Woyke

Zweite Prüferin: Prof. Dr. Birgit Bertelsmeier

Abgabetermin: 15.08.2013

Bearbeitungszeit: 6 Wochen

Gummersbach, August 2013

Inhaltsverzeichnis

1 Abbildungsverzeichnis

2 Abkürzungsverzeichnis

API ..... *Application programming interface*

CSS ..... *Cascading Style Sheets*

GUI ..... *Graphical user Interface*

IDE ..... *Integrated development environment*

JDK ..... *Java Development Kit*

JSON ..... *JavaScript Object Notation*

SDK ..... *Software Development Kit*

XML ..... *Extensible Markup Language*

3 Einleitung

3.1 Ausgangssituation

Dieser Bachelorarbeit geht das Praxisprojekt mit dem Thema „Konzeption und Implementierung eines mobilen SQL-Trainers für das Android-Betriebssystem inklusive Server-Schnittstelle und einer Autovervollständigung bzw. eigenem Keyboard“ voraus, in dem in Zusammenarbeit mit Herrn Patrick Englert der „Mobile SQL-Trainer“ entwickelt wurde. Diese Anwendung wurde nativ auf dem Google-Betriebssystem Android entwickelt. Nun gibt es plattformunabhängige Systeme zur App-Entwicklung wie das Titanium SDK oder das PhoneGap-Framework. Der Entwicklung auf einem plattformunabhängigen System steht die native Entwicklungsform gegenüber. Zum einen verspricht die plattformunabhängige Entwicklungsform natürlich den Vorteil, die Anwendung nach der Entwicklung auch auf anderen Betriebssystemen einsetzen zu können und nicht für jedes Betriebssystem eine eigene Anwendung entwickeln zu müssen. Zum anderen ist fraglich, ob der Entwicklungsaufwand höher ist, ob die Funktionalitäten ähnlich entwickelt werden können und inwieweit der Code angepasst werden muss, damit er auf verschiedenen Geräten ausgeführt werden kann. In dieser Bachelorarbeit möchte ich die native Entwicklungsmethode auf dem Android-Betriebssystem mit der plattformunabhängigen Entwicklungsart mit dem Titanium SDK in diesen Punkten vergleichen.

3.2 Erwartungen

Beim Titanium SDK handelt es sich um eine Entwicklungsumgebung für mobile Anwendungen. Die Anwendungen können laut den Entwicklern für Android, iOS, Tizen, Windows Phone (Windows 8 in Zukunft[1]), BlackBerry sowie für Browseransichten in HTML5 entwickelt werden (Abbildung 1 - Betriebssysteme).

Die Entwickler des Titanium SDKs versprechen auf ihrer Internetseite[2] um zwanzig Prozent verkürzte Entwicklungszeiten gegenüber der nativen Entwicklung („Develop native applications 20% faster than writing in the native language“) und eine Quellcode-Wiederverwendbarkeitsrate

[1]Vgl.: http://developer.appcelerator.com/blog/2013/01/titanium-support-plans-for-windows-8.html [04.07.2013]
[2]Vgl.: http://www.appcelerator.com/platform/titanium-sdk/ [05.07.2013]

von 60-90% („Reuse 60%-90% of developed code when supporting multiple platforms“). Es ist fraglich, ob sich die Entwicklungszeit für einen geschulten Android-Entwickler, der sich in die neue Entwicklungsart in JavaScript einarbeiten muss, um 20% verkürzt.

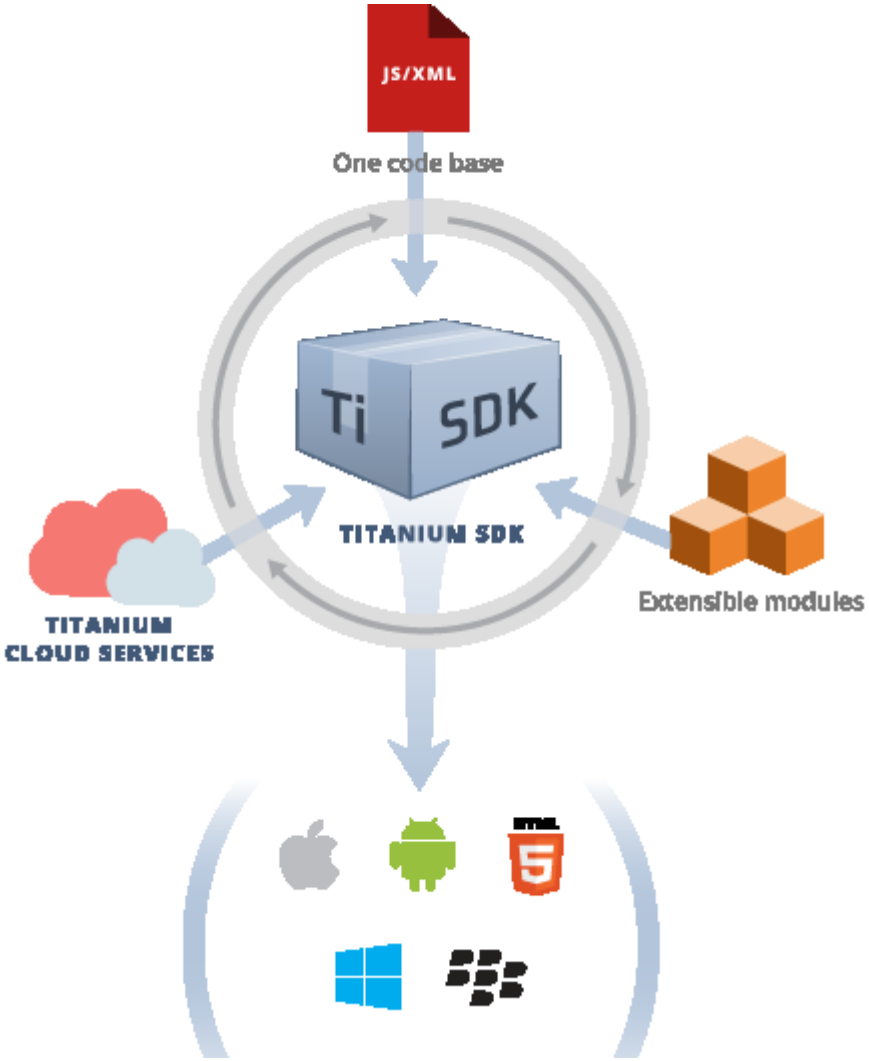

Abbildung 1 - Betriebssysteme

[3]Da das Titanium SDK plattformunabhängige Elemente bereitstellt ist es weiterhin fraglich, ob diese Elemente dieselben Funktionalitäten bereitstellen wie native Android-Elemente. Auch ist es wichtig, auf Gerätefunktionen zugreifen zu können. Bei der nativen Entwicklung auf Android ist dies direkt möglich. Das Titanium SDK muss jedoch plattformunabhängig auf Gerätefunktionen zugreifen können, die natürlich von Hersteller zu Hersteller verschieden sind.

Des Weiteren ist es bei der nativen Entwicklung einer iOS-Anwendung notwendig, diese auf einem Mac-Gerät mit Xcode und dem Cocoa-Touch-Framework[4] zu entwickeln, wobei eine Apple Developer Lizenz benötigt wird, wenn auf einem reellen Gerät getestet oder die Anwendung

[3]Vgl.: http://www.appcelerator.com/platform/titanium-sdk/ [05.07.2013]
[4]Vgl.: https://developer.apple.com/technologies/ios/cocoa-touch.html [05.07.2013]

veröffentlicht werden soll. Fraglich ist, ob darauf bei der plattformunabhängigen Entwicklungsart mit dem Titanium SDK verzichtet werden kann.

3.3 Vorbereitungen

Durch die beiden Wahlpflicht-Module „Entwicklung mobiler Anwendungen“ und „Entwicklung von Apps für Smartphones und Tablets“ sind bereits Grund- und weiterführende Kenntnisse in der nativen App-Entwicklung auf dem Android-Betriebssystem vorhanden. Als Vorbereitung auf diese Arbeit müssen also noch notwendige Fähigkeiten in der Entwicklung mit dem Titanium SDK und JavaScript erlangt werden.

Um einen direkten Vergleich zwischen den beiden Entwicklungsvarianten zu haben, ist es notwendig, eine Anwendung sowohl nativ auf dem Android-Betriebssystem als auch plattformunabhängig mit dem Titanium SDK zu entwickeln. Als Vergleichsobjekt dient die im Praxisprojekt nativ entwickelte Anwendung „Mobile SQL-Trainer“. Diese Anwendung werde ich in den Kernfunktionen im Laufe dieser Arbeit mit dem Titanium SDK entwickeln und die zu prüfenden Vergleichspunkte dabei genauer betrachten.

4 Analyse der nativen Entwicklung auf dem Android-Betriebssystem

4.1 Entwicklungsvorbereitung

Native Android-Anwendungen werden in Java entwickelt. Eine mögliche Entwicklungsumgebung für Java und Android ist Eclipse[5]. Eclipse stellt neben einem Package Explorer, der alle Dateien eines Projekts anzeigt, einer Konsole, einer LogCat (Android-Konsole für Debug-Meldungen) und vielen weiteren nützlichen Ansichten zur Anwendungs-entwicklung einen intelligenten Code-Editor bereit, der direkt während der Programmierung verschiedene Arten von Fehlern feststellen kann und diese dem Entwickler mitteilt. Des Weiteren unterstützt der Code-Editor die Syntaxhervorhebung und weitere Funktionen, die die Entwicklung erleichtern. Eine Beispielfunktion ist das Refactoring, mit Hilfe dessen die Umbenennung einer Methode möglich ist, wobei automatisch alle entsprechenden Methodenaufrufe abgeändert werden. Weiterhin unterstützt Eclipse die gemeinsame Entwicklung einer Anwendung mit Hilfe einer Repository. Neben der Entwicklungsumgebung wird für eine Android-Anwendung das Java JDK und das Android SDK benötigt. Die beiden Developer-Kits stellen die notwendigen Java- und Android-Programmierschnittstellen sowie die Entwicklungstools bereit. Als Online-Hilfestellung steht bei der nativen Android-Entwicklung die Android-Entwickler-Plattform (Android Developers[6]) bereit. Im Android SDK ist der AVD-Manager (Android Virtual Device - Manager) enthalten, mit dem sich sowohl reelle Android-Geräte als auch virtuelle Android-Geräte (Emulatoren) verwalten lassen. Innerhalb des AVD-Managers stehen verschiedene Emulatoren mit mehreren verschiedenen

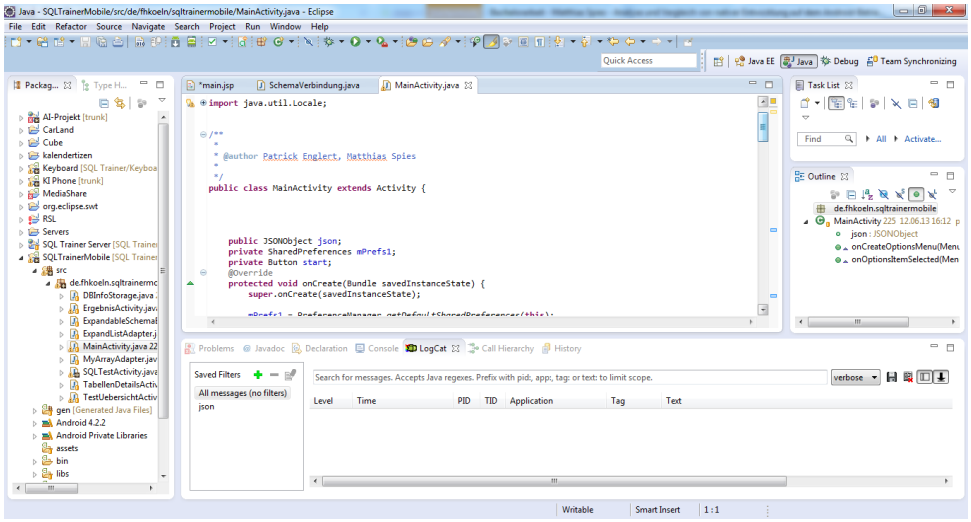

Abbildung 2 - Eclipse Entwicklungsumgebung

[5] Vgl.: http://www.eclipse.org/ [07.07.2013]
[6] Vgl.: http://developer.android.com/index.html [07.07.2013]

Android-Versionen zum Testen einer Anwendung zur Verfügung. Nach der Auswahl eines Geräts oder Emulators kann eine entwickelte Anwendung direkt auf dem reellen oder dem virtuellen Gerät ausgeführt werden. Der erste Start eines Emulators dauert hardwareabhängig mehrere Minuten, weshalb es vorteilhaft ist, dass Eclipse fastDev[7] unterstützt und der Emulator nicht für jeden Anwendungstest erneut gestartet werden muss. Das SDK installiert die Anwendung automatisch und startet sie auf dem Gerät. Während die Anwendung ausgeführt wird, werden Informations- und Debug-Meldungen in der LogCat ausgegeben.

4.2 Entwicklungsstil

Die meisten Anwendungen auf mobilen Geräten haben eine Benutzeroberfläche (GUI). Die Oberflächen der GUI lassen sich ähnlich wie mit dem Window Builder (Eclipse Plugin zur Gestaltung der graphischen Benutzeroberfläche) simpel per Drag&Drop gestalten (Abbildung 3 - Gestaltung der Android-GUI).

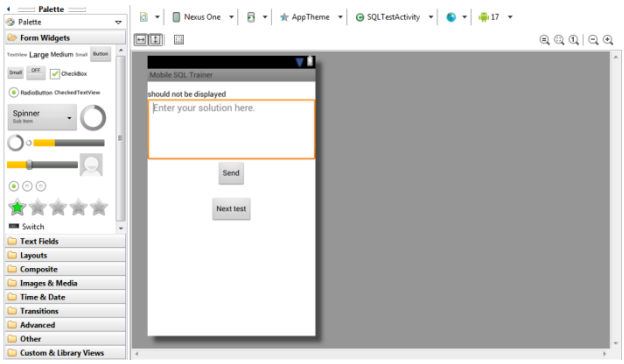

Abbildung 3 - Gestaltung der Android-GUI

Zusätzlich zur Drag&Drop-Variante lässt sich die Oberfläche auch per XML-Code gestalten. Bei XML handelt es sich um eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten. Innerhalb des XML-Codes werden verschiedene Attribute zur Festlegung der Elementeigenschaften genutzt. Zur Formatierung der Oberfläche nutzt man Layouts (im Beispiel ein LinearLayout[8]). Mit Layouts legt man nicht sichtbare Rahmen fest, in denen sich

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/ExpandableschemaBrowserTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/dummy"
        android:textAppearance="?android:attr/textAppearanceLarge" />

            <ExpandableListView
                android:id="@+id/ExpList"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:groupIndicator="@null" />

</LinearLayout>
```

[7] Vgl.: http://developer.appcelerator.com/blog/2011/05/titanium-mobile-intro-series-fastdev-for-android.html [08.07.2013]
[8] Vgl.: http://developer.android.com/reference/android/widget/LinearLayout.html [10.07.2013]

die Kind-Elemente befinden. Mit „android:id“ wird die ID des Elements festgelegt, worüber aus der Activity auf das Element zugegriffen werden kann. Über „android:layout_width“ und „android:layout_height“ wird die Größe des Elements festgelegt. Die Größe eines Elements kann durch absolute oder relative Werte festgelegt werden. „fill_parent“ übernimmt den Wert des übergeordneten Elements, „wrap_content“ wählt die minimale Größe aus, mit der der Inhalt noch angezeigt werden kann. Durch „android:text“ kann der Text, der in diesem Element ausgegeben wird, festgelegt werden. Dabei kann entweder direkt ein String eingegeben, oder eine Referenz auf einen String in der strings.xml eingesetzt werden. Die strings.xml dient dazu, Strings zentral bearbeiten zu können und bei Bedarf eine mehrsprachige Anwendung zu entwickeln, indem mehrere XML-Dateien mit Strings eingesetzt werden. Nach der graphischen Gestaltung der Benutzeroberflächen müssen die Funktionalitäten in Java entwickelt werden. Der Ausführungsteil der Android-Anwendung muss von der Android-Klasse „Activity“ erben (*public class MainActivity extends Activity*). Um dieser nun erschaffenen „Activity“ die Benutzeroberfläche zuzuweisen benutzt man die Methode „setContentView“ (*setContentView(R.layout.activity_main)*). Um nun noch auf die Eigenschaften der graphischen Elemente zugreifen und Events abfangen zu können, kann man die Elemente mit der Methode „findViewById“ suchen. Als View bezeichnet man in der Android-Entwicklung ein graphisches Element auf der Benutzeroberfläche.
Hier ein Beispiel, um auf einen Button zuzugreifen: *(Button) findViewById(R.id.start)*
Um auf ein Event eines Elements zu reagieren, muss man ihm einen „Listener“ hinzufügen. Beim „Listener“ handelt es sich um ein Objekt, welches auf Events wartet und eine bestimmte

Reaktion ausführt. Ein Event-Beispiel wäre eine Betätigung eines Buttons. Diese würde durch einen OnClickListener abgefangen.

```
start = (Button) findViewById(R.id.start);
start.setOnClickListener( new View.OnClickListener()
      {
      public void onClick(View v)
            {
            // Aktion, die bei Betätigung des Buttons ausgeführt wird
            }
      });
```

In diesem Beispiel wird zuerst einer Referenz (start) der Button R.id.start zugewiesen. Anschließend wird ein neuer OnClickListener erstellt und dem Button zugewiesen. Innerhalb des OnClickListeners wird die Operation definiert, die ausgeführt wird, sobald das Event „onClick“

(Betätigung des Buttons) abgefangen wird. Ein weiteres Android-spezifisches Element ist der AsyncTask[9]. Hierbei handelt es sich um einen nebenläufigen Prozess. Diesen setzt man beispielsweise ein, um den Benutzer nicht warten zu lassen, während etwas im Hintergrund ausgeführt wird. Ein ähnliches Java-Element wäre der Thread:

```
private class AsyncTaskBeispiel extends AsyncTask{

	@Override
	protected String doInBackground(String... params) {
		//Eigentliche Aufgabe, die im Hintergrund ausgeführt wird
	}

	@Override
	protected void onPostExecute(String result) {
		//Aufgabe, die nach Beendigung der eigentlichen Aufgabe
		//Ausgeführt wird
	}

	@Override
	protected void onPreExecute() {
		//Aufgabe, die vor der eigentlichen Aufgabe durchgeführt wird
		//Beispielsweise Variableninitialisierung
	}
```

Der AsyncTask beinhaltet die eigentlich auszuführende Aufgabe (doInBackground) und kann sowohl eine vorbereitende Aufgabe (doPreExecute) als auch eine nachbereitende Aufgabe (doPostExecute) durchführen. Ein Element, welches die Anwendungsentwicklung auf Android wesentlich vereinfacht ist die Logcat[10]. Bei der Logcat handelt es sich um Android-Logging-System, welches Debug-Meldungen ausgibt. Auf diese Art und Weise kann man Anwendungen auf Android-Geräten testen und gleichzeitig Fehler-, Test- und Ereignismeldungen in der Entwicklungsumgebung ausgeben lassen.

[9]Vgl.: http://developer.android.com/reference/android/os/AsyncTask.html [10.07.2013]
[10]Vgl.: http://developer.android.com/tools/help/logcat.html [10.07.2013]

4.3 Technische Möglichkeiten

Das Android SDK stellt Elemente bereit, die ausschließlich entwickelt wurden, um auf Android-Geräten genutzt zu werden. Des Weiteren stellt das Android SDK API-Programmbibliotheken und Entwickler-Tools zum Entwickeln, Testen und Debuggen von Android-Anwendungen bereit.

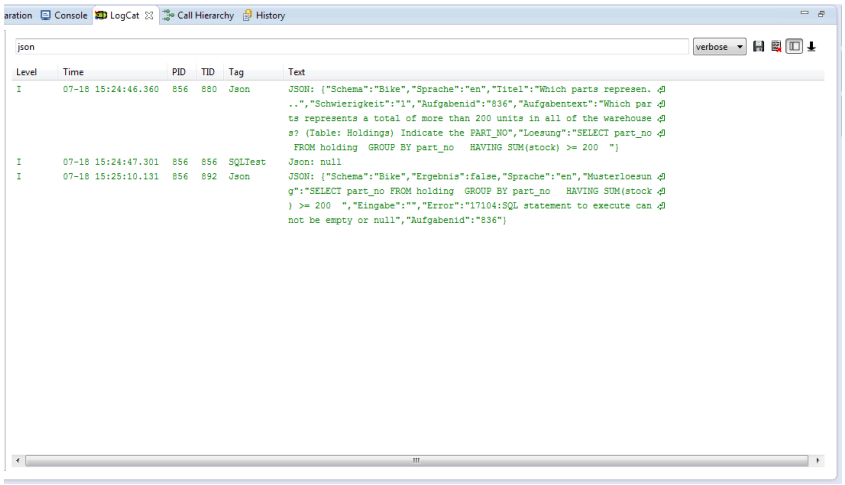

Abbildung 4 - LogCat

Das Android SDK wurde direkt vom Hersteller entwickelt und auf der Android-Entwicklerseite (Android Developers[11]) zur Verfügung gestellt. Dadurch wird der bestmögliche Hardwarezugriff gewährleistet. Der Zugriff auf die Kamera und alle Sensoren der Geräte ist durch das Android SDK möglich. Während der Ausführung einer Anwendung werden Debug- und Info- Meldungen zur laufenden Anwendung in der LogCat[12] angezeigt. Durch die Meldungen fällt das Finden von Fehlern leichter und es wird deutlich, was im Hintergrund ausgeführt wird (Abbildung 4 - LogCat).

4.3.1 Gerätezugriff: Dateisystem, Kamera, Sensoren, Karten, Bluetooth

Um mit dem Android SDK auf die Kamera-Komponente innerhalb der Anwendung zugreifen zu können, steht die *android.hardware.Camera*-Komponente zur Verfügung. Um eine Instanz zu erzeugen verwendet man die *open()*-Methode. Anschließend stehen mehrere Einstellungsmöglichkeiten zur Verfügung: Mit der Methode setParameters() können die Einstellungen bestimmt werden. Die Einstellungen können in der Android Developer-Dokumentation[13] nachgeschaut werden, es handelt sich um eine Vielzahl von Hard- und Softwarekonfigurationen. Neben der Kamera-Komponente kann natürlich auch auf das Android-Dateisystem zugegriffen werden. Um eine Datei abzulegen oder zu öffnen kann sowohl auf das Verzeichnis der Anwendung als auch auf eine Speicherkarte zugegriffen werden. Auf ein Anwendungsverzeichnis einer anderen Anwendung kann aufgrund der Android-Rechtevergabe

[11] Vgl.: http://developer.android.com/sdk/index.html [12.07.2013]
[12] Vgl.: http://developer.android.com/tools/help/logcat.html [12.07.2013]
[13] Vgl.: http://developer.android.com/reference/android/hardware/Camera.Parameters.html [12.07.2013]

nicht zugegriffen werden. Um eine Datei auf dem internen Speicherplatz des Geräts abzulegen, geht man folgendermaßen vor:

File file = new File(context.getFilesDir(), filename);

Hier wird eine Instanz vom Typ java.io.File erstellt und mit *context.getFilesDir()* das Anwendungsverzeichnis als Speicherort festgelegt. Der Dateiname wird durch die Referenz *filename* zugewiesen.

Bevor auf die Speicherkarte zugegriffen werden kann, muss überprüft werden, ob diese vorhanden ist und ob darauf zugegriffen werden kann. Über die Methode

```
Environment.getExternalStorageState();
```

kann der Status der Speicherkarte abgefragt werden. Um anschließend eine Datei (im Beispiel ein Bild) auf der externen Speicherkarte abzulegen, kann folgender Code eingesetzt werden:

```
File file = new File(Environment.getExternalStoragePublicDirectory(
            Environment.DIRECTORY_PICTURES), albumName);
```

Um auf Sensordaten des Geräts zuzugreifen, muss zuerst eine Instanz der Komponente *android.hardware.SensorManager*[14] erzeugt werden:

```
SensorManager mSensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);
```

Um zu überprüfen, ob der anzusprechende Sensor im Gerät existiert, kann man eine Liste mit allen vorhandenen Sensoren erzeugen:

```
SensorManager.getSensorList(Sensor. TYPE_ALL)
```

Anschließend muss eine Referenz auf den Sensor erzeugt und der Listener registriert werden:

```
Sensor accelerometer = SensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
mSensorManager.registerListener(activity, accelerometer,
SensorManager.SENSOR_DELAY_NORMAL);
```

Beim Listener handelt es sich um ein Objekt, welches beim Eintreten eines Zustandes eine Aktion ausführt. In diesem Fall handelt es sich um den Beschleunigungssensor. Nach der Registrierung

[14] Vgl.: http://developer.android.com/reference/android/hardware/SensorManager.html [14.07.2013]

kann ein Event-Listener eingesetzt werden, der auf ein eintretendes Event reagiert. Ein Beispiel-Event wäre eine Sensordaten-Änderung:

```
void onSensorChanged(SensorEvent event)
```

Um Geräteressourcen zu sparen sollten die Event-Listener eingesetzt werden. Wenn die Sensordaten dauerhaft abgefragt werden würden, würde dies den Akku des Geräts schneller entleeren.

Um die Google Maps API V2 [15] innerhalb einer Anwendung zu nutzen, muss zuerst ein API Key generiert werden. Dazu ist es notwendig, das Projekt in der Google API Console zu registrieren. Zusätzlich müssen folgende Zugriffe in der Manifest-Datei zugelassen werden: Internet (`android.permission.INTERNET`), Verbindungsstatus (`android.permission.ACCESS_NETWORK_STATE`), Zugriff auf Googles Web-Dienste (`com.google.android.providers.gsf.permission.READ_GSERVICES`) und Speicherkartenzugriff (`android.permission.WRITE_EXTERNAL_STORAGE`). Um die eigene Position auf der Karte anzuzeigen sind noch weitere Zugriffserlaubnisse erforderlich (`android.permission.ACCESS_COARSE_LOCATION`, `android.permission.ACCESS_FINE_LOCATION`).

Da die Version zwei der Google Maps die zweite Version von OpenGL ES benötigt, muss in der Manifest-Datei Folgendes zusätzlich eingefügt werden:

```
<uses-feature
      android:glEsVersion="0x00020000"
      android:required="true"/>
```

Um eine Karte in eine Anwendung zu integrieren, kann folgender Code in der Layout-XML-Datei eingefügt werden:

```
<?xml version="1.0" encoding="utf-8"?>
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/map"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:name="com.google.android.gms.maps.MapFragment"/>
```

Anschließend erscheint die Karte in der Anwendung.

[15] Vgl.: https://developers.google.com/maps/documentation/android/start [12.07.2013]

Um Bluetooth in einer Anwendung zu nutzen, ist es notwendig, den Zugriff auf das Bluetooth-Modul in der Manifest-Datei zuzulassen:[16]

```
<uses-permission android:name="android.permission.BLUETOOTH" />
```

Anschließend muss geprüft werden, ob das Gerät Bluetooth unterstützt:

```
BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
if (mBluetoothAdapter == null) { // Device does not support Bluetooth }
```

Wenn das Gerät Bluetooth unterstützt, muss überprüft werden, ob Bluetooth bereits aktiviert ist. Falls es noch nicht aktiviert ist, muss es per Intent über die Systemeinstellungen aktiviert werden. Ein Intent ist ein Systemaufruf, um Änderungen durchzuführen oder Anwendungen zu starten. Falls Bluetooth noch nicht aktiviert war, wird der Benutzer aufgefordert, die Bluetooth-Aktivierung zu bestätigen (Abbildung 5 - Bluetooth-Aktivierung). Anschließend kann man alle bereits einmal

```
if (!mBluetoothAdapter.isEnabled()) {
Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
}
```

verbundenen Geräte in einer Liste verwalten und bei Bedarf dem Benutzer in einer ListView zur Auswahl geben, um sich erneut mit einem dieser Geräte zu verbinden.

Um neue Bluetooth-Geräte in Reichweite zu finden, wird die Methode *startDiscovery ()* der Komponente *BluetoothAdapter*[17] verwendet. Die Suche dauert 12 Sekunden. Erst anschließend werden die Namen der Bluetooth-Geräte festgestellt. Über einen *BroadcastReceiver* werden Name und Mac-Adresse der Geräte festgestellt. Zum Verbinden mit einem anderen Gerät wird nur die Mac-Adresse benötigt. Um das Gerät für andere Bluetooth-Geräte sichtbar zu machen, wird folgender Intent eingesetzt:

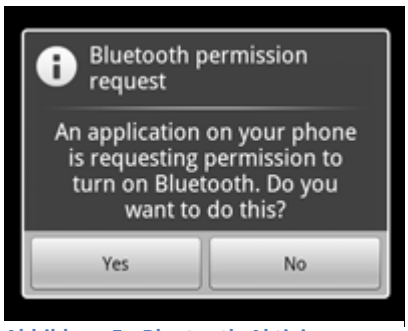

Abbildung 5 - Bluetooth-Aktivierung

[16]Vgl.: http://developer.android.com/guide/topics/connectivity/bluetooth.html [15.07.2013]
[17]Vgl.: http://developer.android.com/reference/android/bluetooth/BluetoothAdapter.html#startDiscovery()[15.07.2013]

```
Intent discoverableIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
discoverableIntent.putExtra(BluetoothAdapter.EXTRA_DISCOVERABLE_DURATION,
300);
startActivity(discoverableIntent);
```

Hier wird auch die Standart-Sichtbarkeits-Dauer von 120 Sekunden auf 300 Sekunden erhöht. Die Erlaubnis zum Aktivieren des Bluetooth-Adapters zur Sichtbarkeit von 300 Sekunden wird wieder vom Benutzer per Dialog eingeholt (Abbildung 6 - Bluetooth-Sichtbarkeit). Um nun eine Verbindung zwischen zwei Bluetooth-Geräten herzustellen, muss das eine Gerät den Server stellen und das andere Gerät sich als Client verbinden. Um anschließend mit dem jeweils anderen Gerät zu kommunizieren, muss ein *InputStream*[18] und ein *OutputStream*[19] verwendet werden. In den *OutputStream* wird mit der Methode write() geschrieben, um dem anderen Gerät etwas mitzuteilen, aus dem *InputStream* wird mit *read()* die Nachricht des anderen Geräts gelesen.

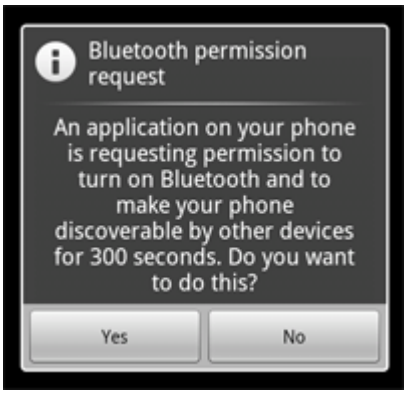

Abbildung 6 - Bluetooth-Sichtbarkeit

4.3.2 Medienwiedergabe und -aufnahme:

Um Medien wiedergeben zu können stehen im Android SDK mehrere Möglichkeiten zur Verfügung. Mediendateien, also Audio- und Videodateien sowie Bilddateien, können sowohl aus dem RAW-Verzeichnis der Anwendung als auch aus dem Dateisystem geöffnet werden. Zusätzlich kann der Android

```
Set<BluetoothDevice> pairedDevices = mBluetoothAdapter.getBondedDevices();
// Falls vertraute Geräte vorhanden sind
if (pairedDevices.size() > 0) {
    for (BluetoothDevice device : pairedDevices) {
        mArrayAdapter.add(device.getName() + "\n" + device.getAddress());
    }
}
```

MediaPlayer[20] Medien aus einem Daten-Stream über das Netzwerk wiedergeben. Eine Mediendatei aus dem RAW-Verzeichnis der Anwendung kann sehr simpel wiedergegeben werden:

[18] Vgl.: http://developer.android.com/reference/java/io/InputStream.html [17.07.2013]
[19] Vgl.: http://developer.android.com/reference/java/io/OutputStream.html [17.07.2013]
[20] Vgl.: http://developer.android.com/reference/android/media/MediaPlayer.html [17.07.2013]

```
MediaPlayer mediaPlayer = MediaPlayer.create(context, R.raw.sound_file_1);
mediaPlayer.start();
```

Der Android MediaPlayer unterstützt eine Vielzahl von Medienformaten, unter anderem die gängigsten Bildformate (JPEG, BMP, PNG, GIF), gängige Videoformate (3GP, MPEG4, MKV) und gängige Musikformate (MP3, WAV, MIDI).

Um Audioaufnahmen zu erstellen wird der Android MediaRecorder[21]genutzt. Um eine Aufnahme anzufertigen ist es notwendig eine Instanz von *android.media.MediaRecorder* zu erzeugen und die Aufnahmequelle (meist das Gerätemikrofon) zuzuweisen:

```
MediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC)
```

Anschließend muss das Ausgabeformat und die Ausgabedatei festgelegt werden:

MediaRecorder.setOutputFormat()
MediaRecorder.setOutputFile()

Danach muss der Audio-Encoder festgelegt und der MediaRecorder vorbereitet werden:

```
MediaRecorder.setAudioEncoder()
MediaRecorder.prepare()
```

Nun ist der MediaPlayer bereit zur Aufnahme. Die Aufnahme wird begonnen durch *MediaRecorder.start()* und beendet durch *MediaRecorder.stop()*.

4.3.3 Animationen

Mit dem Android SDK gibt es zwei Möglichkeiten um eine Animation zu erstellen: Zum einen ist es möglich, die Eigenschaften eines Objekts über einen Zeitraum zu verändern (Property-Animation) und zum anderen kann eine ganze View animiert werden (View-Animation). Mit der Eigenschaften-Animation können alle Elemente animiert werden. Mit der View-Animation können nur Views animiert werden. Bei der Eigenschaften-Animation muss der zu verändernde Wert (x,y,alpha) und ein Endwert eingegeben werden (ObjectAnimator[22]). Hierbei kann es sich um einen Integer-Wert (Positions- oder Größentransformation) oder einen Farbwert (Hintergrundfarbe, Alpha-Wert) handeln. Optional kann auch ein Startwert angegeben werden. Wird kein Startwert eingegeben, wird der Wert des Elements über die *get*-Methode bezogen. Weiterhin kann eine Animationsdauer und eine Wiederholungsart ausgewählt werden. Mit dieser Animationsart werden die Werte des Objekts tatsächlich geändert und können über die get-Methoden abgefragt

[21]Vgl.: http://developer.android.com/reference/android/media/MediaRecorder.html [17.07.2013]
[22]Vgl.: http://developer.android.com/reference/android/animation/ObjectAnimator.html [17.07.2013]

werden. Die View-Animationen unterscheiden sich in Form von zwei Untertypen: Zum einen die „Tween Animation“ und zum anderen die „Frame Animation“. Die „Tween Animation“ verändert ähnlich wie die Property-Animation die Werte einer View. Hiermit können Drehungen, Positions- und Größenänderungen sowie Ein- und Ausblendungen durchgeführt werden. Die „Frame Animation“ zeigt eine Sequenz von Bildern ähnlich wie ein Video an. Hierbei muss festgelegt werden, welche Bilder und wie lange diese jeweils angezeigt werden sollen. Der Einsatz von View-Animation ist mit einem geringeren Aufwand verbunden als der Einsatz von Property-Animationen, allerdings werden bei Property-Animationen die Werte tatsächlich verändert. Dies hat beispielsweise zum Vorteil, dass ein verschobener Button weiterhin benutzbar bleibt. Bei der View-Animation müsste weiterhin an die vorherige Stelle geklickt werden, um den Button zu betätigen.

4.3.4 Open-GL

Das Android SDK unterstützt 2D – und 3D – Grafiken mit der Open Graphics Library (OpenGL[23]). Bei OpenGL handelt es sich um eine Spezifikation für eine Programmierschnittstelle zur Entwicklung von 2D – und 3D – Grafiken. Um OpenGL in einer Android-Anwendung zu verwenden, ist es notwendig, die Klassen GLSurfaceView[24] und GLSurfaceViewRenderer in der Activity zu implementieren. Um Touch-Screen-Events abzufangen, ist es notwendig, dass die Klasse GLSurfaceView vererbt wird. Bei der Klasse GLSurfaceView.Renederer handelt es sich um das Interface, welches die Methoden beinhaltet, die benötigt werden, um Grafiken zu animieren. Um diese Methoden nutzen zu können, ist es notwendig, eine zusätzliche Klasse zu erstellen, die dieses Interface implementiert und auf diese Klasse mit der Methode GLSurfaceView.setRenderer() aus der eigentlichen Activity zuzugreifen. Um das Interface zu verwenden, müssen die Methoden onSurfaceCreated(), onDrawFrame() und onSurfaceChanged() implementiert werden. Bei der onSurfaceCreated()-Methode

Abbildung 7 - Android OpenGL-Beispiel

[23]Vgl.: http://www.opengl.org/about/ [18.07.2013]
[24]Vgl.: http://developer.android.com/reference/android/opengl/GLSurfaceView.html [18.07.2013]

handelt es sich um eine Initialisierungsmethode, die nur einmal aufgerufen wird. Die Methode onDrawFrame() ist die Hauptmethode, die bei jedem Neuzeichnen der Grafik aufgerufen wird. Die Methode onSurfaceChanged() wird aufgerufen, wenn die View neu gezeichnet wird. Dies passiert hauptsächlich, wenn das Gerätedisplay gedreht und die Orientierung von „landscape“ zu „portrait“ wechselt (oder andersrum). Nach der Implementierung des Interfaces und der Vererbung der Klasse GLSurfaceView kann auf die „OpenGL ES 1.0 API“-, „OpenGL ES 2.0 API“-, und „OpenGL ES 3.0 API“- Schnittstellen zugegriffen werden. Da die Verwendung von OpenGL sehr komplex ist, stellt das Android-Developer-Portal Lerneinheiten zu OpenGL bereit[25].

4.3.5 Externe Libraries

Bei der Entwicklung für Android ist es möglich, externe Libraries einzusetzen. Bei den Libraries handelt es sich um vorgefertigte Anwendungsteile, auf die über eine Schnittstelle zugegriffen werden kann. Durch den Einsatz von externen Libraries müssen diese Anwendungsteile nicht selbst entwickelt werden, sondern können direkt benutzt werden. Für die Android-Entwicklung steht kein zentrales Portal für die Libraries bereit, viele Entwickler bieten diese auf ihren eigenen Seiten an. Die importierten Libraries müssen in das „libs“- Verzeichnis der Anwendung eingefügt und anschließend zum „Build Path“ hinzugefügt werden, um auf diese Library zugreifen zu können. Außerdem wird diese so bei der Kompilierung mit in die Anwendung einbezogen. Die Libraries werden meist als .jar-File angeboten.

Google stellt mit dem Android Support Library Package selbst eine Sammlung von hilfreichen Libraries zur Verfügung. Die Libraries sind nach der Version, ab der sie eingesetzt werden können, benannt (V4 ab API-Level 4). Die Libraries enthalten sowohl Elemente zur optischen Gestaltung der Anwendung als auch interne Funktionen, wie String-Funktionen.

[25] Vgl.: http://developer.android.com/training/graphics/opengl/index.html [20.07.2013]

4.3.6 Google Cloud Messaging for Android[26]

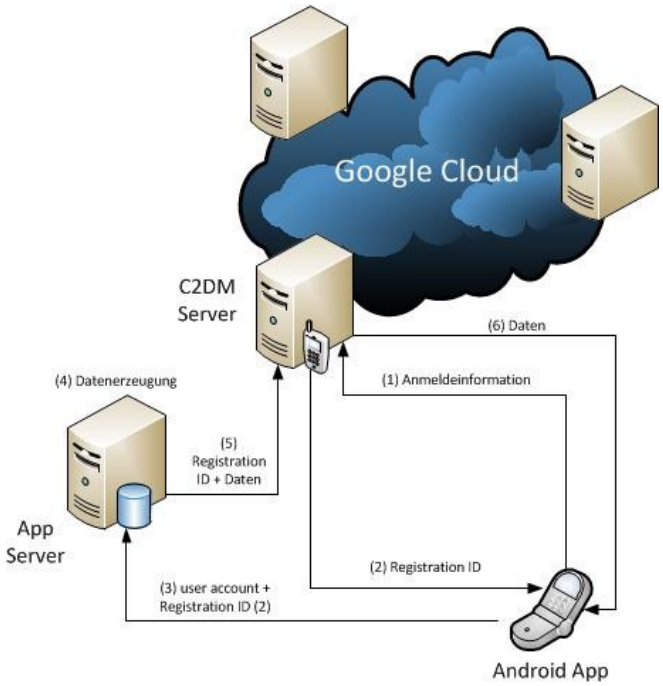

Abbildung 8 - Google Cloud Services

Google stellt mit dem Google Cloud Messaging einen Service zur Verfügung, mit dem Daten vom Server an Android-Geräte gesendet und von Android-Geräten empfangen werden können. Dadurch lassen sich Bild-Uploads, Push-Notifikationen und viele andere Cloud-Dienste realisieren. Zur Realisierung von Push-Notifikationen steht das „Android Cloud to Device Messaging Framework“ zur Verfügung. Dabei registriert sich die Anwendung bei dem entsprechenden Cloud-to-Device-Server und erhält eine Geräte-Registrations-ID. Die Push-Notifikationen werden über den Server durch den Google Play-Store an das Gerät mit der entsprechenden Geräte-Registrations-ID gesendet. Des Weiteren werden Push-Notifikationen bei abgeschaltetem Gerät zugestellt, wenn sich das Gerät wieder mit dem Server verbindet. Die Push-Notifikationen können als Nachricht nur 1024 Bytes enthalten und sollten deshalb kurz- gehalten werden. Die Grafik zeigt die Funktion des Google Cloud Services (Abbildung 8 - Google Cloud Services).

Abbildung 9 - Mobile SQL-Trainer (Android)

4.4 Beispielanwendung „Mobile SQL-Trainer“ (native Android-Entwicklung)

Um eine direkte Vergleichsmöglichkeit zwischen nativer Entwicklung und plattformunabhängiger Entwicklung zu haben, wurde im Praxisprojekt die Anwendung „Mobile SQL-Trainer“ nativ auf Android entwickelt. Diese Anwendung wurde mit der Entwicklungsplattform Eclipse entwickelt. Die Datenbankverwaltung wird dabei von einer Serverschnittstelle übernommen und braucht nicht in der mobilen Anwendung entwickelt

[26] Vgl.: https://developer.android.com/google/gcm/index.html [20.07.2013]

werden. In der mobilen Anwendung brauchen nur interne Funktionen und die Kommunikationsmöglichkeit zur Serverschnittstelle implementiert werden.

Beim Aufruf der Anwendung erscheint der Auswahlbildschirm der Testeinstellungen (Abbildung 9 - Mobile SQL-Trainer (Android)). Hier erkennt man die nativen Android-Elemente, die sehr simpel über das dem Window-Builder ähnlichen Drag&Drop-Tool eingefügt und positioniert werden können. Bei den Elementen handelt es sich um TextViews, einen Button und Spinner zur Auswahl der Einstellungen. Nach der Auswahl der Einstellungen gelangt man über den „Test starten“- Button auf die Oberfläche der ersten Testaufgabe (Abbildung 10 - Mobile SQL-Trainer: Aufgabe (native Version)).

Abbildung 10 - Mobile SQL-Trainer: Aufgabe (native Version)

In dieser Oberfläche wurden eine TextView, zwei Buttons und eine MultiAutoCompleteTextView verwendet. Alle Buttons haben einen OnClickListener (Abbildung 11 - Aufgabe gelöst), der die entsprechende Aktion ausführt. Die Anfragen an die Serverschnittstelle für das Laden der Aufgabe, das Überprüfen der Lösung und das Laden der Schemata und Tabellen sowie Tabelleninhalte für den Schema-Browser werden über die Android HttpGet-Komponente[27] realisiert. Die Antworten, die allesamt im JSON-Format entgegengenommen werden, können durch die Java-eigenen Formate *JSONObject* und *JSONArray* simpel verarbeitet werden. Nach einem Lösungsversuch und der Betätigung des „Abschicken“-Buttons wird die Lösung durch die Kommunikation mit der Serverschnittstelle verifiziert und das Ergebnis der Überprüfung zusammen mit dem Fehlercode (bei nicht korrekt gelöster Aufgabe) ausgegeben (Abbildung 11 - Aufgabe gelöst). Bei einer korrekt gelösten Aufgabe besteht nur die Möglichkeit, die nächste Aufgabe zu bearbeiten. Wird eine Aufgabe falsch

Abbildung 11 - Aufgabe gelöst

[27] Vgl.: http://developer.android.com/reference/org/apache/http/client/methods/HttpGet.html [20.07.2013]

Abbildung 13 - Schema Browser

Abbildung 12 - Tabelleninhalte

beantwortet, besteht zum einen die Möglichkeit, die Aufgabe erneut zu beantworten und zum anderen die Möglichkeit, die Aufgabe zu überspringen und zur nächsten Aufgabe zu gelangen. Während der Bearbeitung der Aufgaben ist es natürlich notwendig, die Datenbank-Struktur zu kennen. Dafür wurde ein Schema-Browser implementiert. Den Schema-Browser (Abbildung 13 - Schema Browser) können über die Android-Hardware-Menü-Taste (bei neueren Geräten der Software-Menü-Button) aufrufen werden.

Im Schema-Browser wurde zum Auflisten der Tabellen die Android-Komponente ExpandableListView benutzt und durch einen von der Klasse *BaseExpandableListAdapter* erbenden Adapter auf die Bedürfnisse des Schema-Browsers angepasst. Zusätzlich wurde eine TextView eingefügt, die den Namen des Schemas ausgibt. Um den Schema-Browser wieder zu schließen nutzt man den Android-Back-Button (Software/Hardware). Zudem wurde eine Tabelleninhaltsansicht implementiert, welche durch die Berührung eines Spaltennamens aufgerufen wird und die Spaltennamen und Inhalte der ausgewählten Tabelle anzeigt (Abbildung 12 - Tabelleninhalte). Da manche Tabellen aufgrund ihrer Größe nicht auf einem Bildschirm angezeigt werden können, ist es notwendig, in der Tabelle sowohl horizontal als auch vertikal scrollen zu können. Zur Realisierung der Scroll-Möglichkeit wurde eine *HorizonalScrollView*[28] in eine *ScrollView*[29] verschachtelt. Darin befindet sich ein *TableLayout*[30] zur Darstellung der Tabelleninhalte. Die Daten im *TableLayout* werden aus dem JSON-Objekt extrahiert, welches durch eine Anfrage an die Serverschnittstelle erhalten wurde.

Wenn alle gestellten Aufgaben bearbeitet wurden, gibt die Anwendung ein Gesamtergebnis aus (Abbildung 14 - Gesamtergebnis (nativ)).

[28] Vgl.: http://developer.android.com/reference/android/widget/HorizontalScrollView.html [22.07.2013]
[29] Vgl.: http://developer.android.com/reference/android/widget/ScrollView.html [22.07.2013]
[30] Vgl.: http://developer.android.com/reference/android/widget/TableLayout.html [22.07.2013]

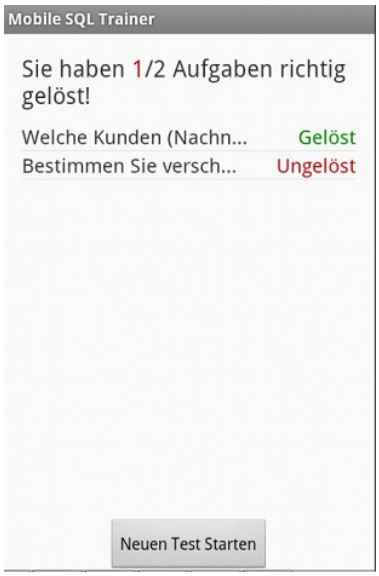

Abbildung 14 - Gesamtergebnis (nativ)

Die Ergebnisse der einzelnen Lösungen werden dann in einer *ListView*[31] ausgegeben. Für die Anordnung innerhalb einer Zeile einer *ListView* muss ein eigener *ArrayAdapter*[32] erzeugt werden. Durch den Adapter können Android-Elemente, wie zum Beispiel *TextView* 's[33] innerhalb einer *ListView*-Zeile eingefügt werden. Zusätzlich können innerhalb der Zeile Events abgefangen werden (beispielsweise durch einen Button). Des Weiteren ist eine *ArrayList*[34] zur Befüllung der *ListView*-Zeilen notwendig. Innerhalb der Zeilen werden der Titel der Aufgabe und das Ergebnis des Lösungsversuchs ausgegeben. Zusätzlich wurde eine Anzeige implementiert, die den Anteil der richtig gelösten Aufgaben ausgibt. Weiterhin beinhaltet diese Oberfläche einen Button, der den Test beendet und einen neuen Test beginnt. Um einen Test abzubrechen und auf die Startoberfläche zu gelangen wird der Android-Back-Button genutzt.

[31]Vgl.: http://developer.android.com/guide/topics/ui/layout/listview.html [22.07.2013]
[32]Vgl.: http://developer.android.com/reference/android/widget/ArrayAdapter.html [22.07.2013]
[33]Vgl.: http://developer.android.com/reference/android/widget/TextView.html [22.07.2013]
[34]Vgl.: http://developer.android.com/reference/java/util/ArrayList.html [22.07.2013]

5 Analyse der plattformunabhängigen Entwicklung mit dem Titanium SDK

5.1 Entwicklungsvorbereitung

Im Titanium SDK ist das Titanium Studio enthalten. Hierbei handelt es sich um die Entwicklungsumgebung für die plattformunabhängigen Anwendungen.

Das Titanium Studio benötigt das Java JDK. Allerdings wird unbedingt das JDK 1.6 mit JRE benötigt. Wenn also ein anderes JDK installiert ist, muss dieses vorher deinstalliert und die 1.6-Version installiert werden. Das Titanium Studio benötigt zusätzlich das jeweilige Plattform SDK, also im Falle von Android das Android SDK, allerdings ist es hier wichtig, dass der Pfad zum jeweiligen SDK kein Leerzeichen enthält, sonst wird er von Titanium Studio nicht erkannt. Zusätzlich ist es wichtig, dass die richtige Version des SDKs und des Emulators/Simulators installiert wird. Bei der Entwicklung für Android ist die API Version 7 (Android 2.1) notwendig, anderenfalls wird das Android SDK nicht erkannt. Die Umgebungsvariablen müssen manuell eingetragen werden. Hierbei muss zum einen der Pfad zum Java JDK ergänzt und zum anderen der Pfad zum Android SDK eingetragen werden.

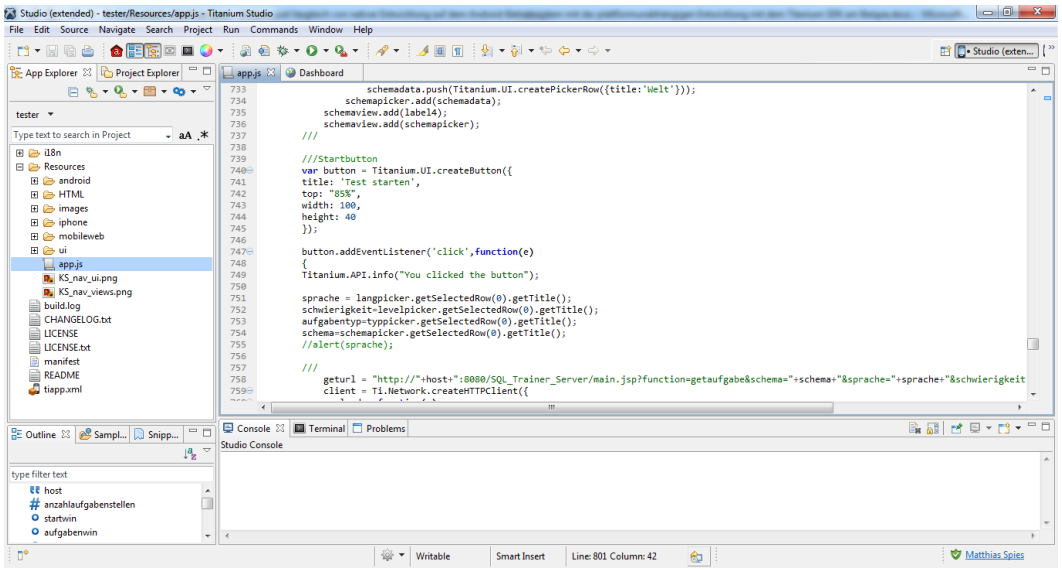

Abbildung 15 - Titanium Studio

Das Titanium SDK stellt das Beispielprojekt „Kitchen Sink“ zur Verfügung, mit Hilfe dessen der Einstig in die Entwicklung besser gelingt und Funktionen getestet werden können. Weiterhin basiert das Titanium Studio auf Aptana[35] und somit auf Eclipse[36], was nicht verwunderlich ist, wenn man die Entwickleroberfläche betrachtet(Abbildung 15 - Titanium Studio).

[35] Vgl.: http://www.aptana.com/ [22.07.2013]

5.2 Entwicklungsstil

5.2.1 Die traditionelle Entwicklungsmethode

Anwendungen mit dem Titanium SDK werden in JavaScript entwickelt. Die graphische Oberfläche einer Anwendung muss mit dem Titanium SDK über den Code entwickelt werden, es wird kein Drag&Drop - Tool zur Oberflächengestaltung zur Verfügung gestellt. Allerdings müssen nicht alle Elemente selbst erstellt werden, da Titanium vorgefertigte Elemente zur Verfügung stellt. Um eine Oberfläche zu erstellen verwendet man den Befehl *Titanium.UI.createWindow()*. Um Schreibarbeit zu sparen kann alternativ zu „Titanium“ die Abkürzung „Ti“ verwendet werden.
In der nachfolgenden geschweiften Klammer können Eigenschaften der Oberfläche direkt festgelegt werden (beispielsweise die Hintergrundfarbe: *backgroundColor:"#fff"*).
Die so erschaffene Oberfläche kann durch sogenannte *Views*[37] eingeteilt werden. Alle Oberflächen und Views einer Anwendung können in einer JavaScript-Datei verwaltet und benutzt werden. Die *Views* werden durch *Titanium.UI.createView()* erstellt und müssen durch oberflaeche.add(view) auf der Oberfläche eingefügt werden („oberflaeche“ und „view“ sind hierbei stellvertretend für die Variablennamen der Elemente). Das Titanium SDK stellt Beschriftungselemente (Label), Eingabefelder (TextArea), Buttons und viele weitere nützliche Elemente zur Verfügung. Zu jedem Element gibt es eine Beschreibung, auf welchen Betriebssystemen diese eingesetzt werden können. Zusätzlich gibt es eine genauere Beschreibung zum Element sowie ein Anwendungsbeispiel in der Titanium Dokumentation[38] (Abbildung 16 - Titanium Online Dokumentation).

[36] Vgl.: http://www.heise.de/developer/meldung/Titanium-Studio-Entwicklungsumgebung-fuer-plattformunabhaengige-Apps-1260783.html [22.07.2013]
[37] Vgl.: http://docs.appcelerator.com/titanium/3.0/#!/api/Titanium.UI.View [22.07.2013]
[38] Vgl.: http://docs.appcelerator.com/titanium/latest/#!/api/Titanium.UI.Label [22.07.2013]

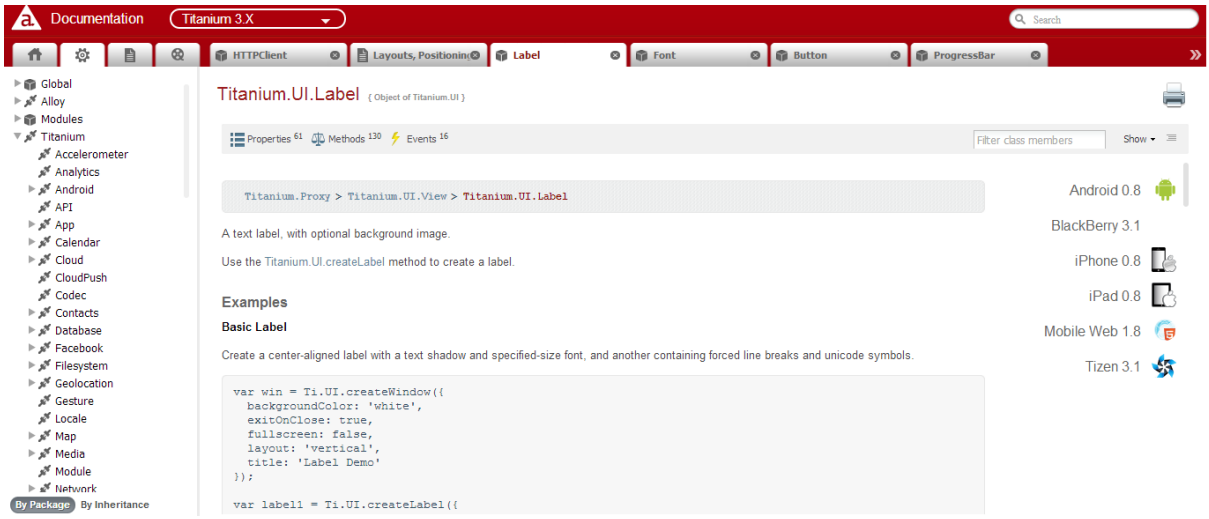

Abbildung 16 - Titanium Online Dokumentation

Erstellte Elemente müssen mit *view.add(element)* in eine *View* oder direkt mit *oberflaeche.add(element)* auf der Oberfläche eingefügt werden. Um eine Oberfläche auf dem Bildschirm anzuzeigen muss diese mit oberflaeche.open() geöffnet werden. Um Elemente und *Views* innerhalb der Oberfläche oder *View*, in der sie eingefügt werden zu positionieren, verwendet man entweder direkt bei der Erstellung oder nachträglich CSS-Tags[39] (Abbildung 17 - CSS-Tags zur Positionierung).

```
78    var schematitelview = Titanium.UI.createView(
79        {
80        width:"100%",
81        height:"20%",
82        backgroundColor:'#9c0c08',
83        top:0,
84        left:0
85        });
```

Abbildung 17 - CSS-Tags zur Positionierung

An diesem Beispiel (Abbildung 17 - CSS-Tags zur Positionierung) wird deutlich, wie man direkt bei der Erstellung Attribute und die Positionierung des Elements (in diesem Beispiel eine View) festlegt:

Mit *width* legt man die Weite des Elements fest. Es können sowohl absolute als auch relative Werte benutzt werden. Relative Werte beziehen sich auf die Elemente, auf denen Sie eingefügt werden, bzw. bei Oberflächen direkt auf die Bildschirmbreite.

[39] Vgl.: http://www.w3school.com/css/ [23.07.2013]

Mit *height* wird die Höhe des Elements festgelegt. Durch *top, left, right, bottom* kann das Element positioniert werden. Zusätzlich existieren viele weitere CSS-Tags, die die Attribute des Elements festlegen.

Um auf Geräteeigenschaften wie Betriebssystem, Version, Höhe und Breite zuzugreifen, können die Titanium-Eigenschaften Titanium.Platform.osname, Titanium.Platform.version, Titanium.Platform.displayCaps.platformHeight und Titanium.Platform.displayCaps.platformWidth abgefragt werden.

5.2.2 Die Entwicklung mit dem Appcelerator Alloy Framework

[40]Seit November 2012 existiert eine weitere Möglichkeit der Anwendungsentwicklung im Titanium Studio: Die Entwicklung mit dem Appcelerator Alloy-Framework. Erst am 19. Februar 2013 wurde die Version 1.0 released. Es handelt sich bei dieser Entwicklungsweise also noch um eine zum jetzigen Stand relativ neue und eventuell noch nicht ausgereifte Technik zur Anwendungsentwicklung. Auch mit dem Alloy-Framework werden plattformunabhängige Anwendungen für die Plattformen Android, iOS, Tizen, BlackBerry und MobileWeb entwickelt. Die Entwicklung mit dem Alloy-Framework erinnert stark an die HTML5-Entwicklung, bei der Benutzeroberfläche, Eigenschaften und Logikteil getrennt sind (Abbildung 18 - Entwicklung mit dem Appcelerator Alloy-Framework). Die Oberfläche wird in der index.xml im Verzeichnis „views“ erstellt. Die Eigenschaften werden in der index.tss-Datei im Verzeichnis „styles“ erstellt und der Logikteil wird in der index.js-Datei im Verzeichnis „controllers“ erstellt. Durch diese Trennung wird

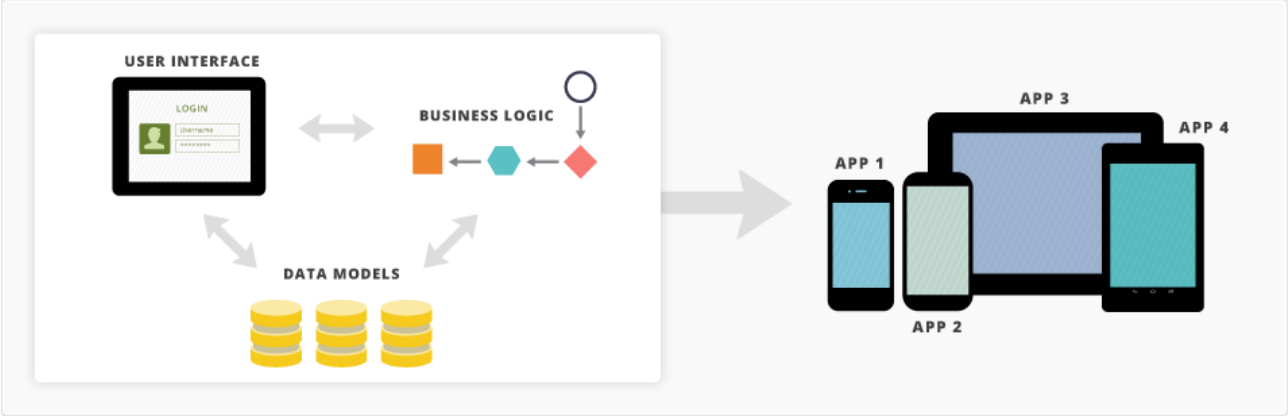

Abbildung 18 - Entwicklung mit dem Appcelerator Alloy-Framework

[40]Vgl.: http://www.appcelerator.com/platform/alloy/ [23.07.2013]

die Entwicklung einer Anwendung übersichtlicher. Zusätzlich versprechen die Entwickler 50-60% Code-Einsparung[41] gegenüber der herkömmlichen Entwicklungsweise. Außerdem soll der Code einfacher lesbar und wiederverwendbar sein.

Die Eigenschaften der Elemente werden in einer tss-Datei verwaltet. Beim tss-Format handelt es sich um das dem css-Format ähnlichem „Titanium Style Sheet“-Format:

```
// Jedes Element mit der Klasse:“container”
".container": {
        backgroundColor:"white"
},
// Alle Labels auf der Oberfläche
"Label": {
        width: Ti.UI.SIZE,
        height: Ti.UI.SIZE,
        color: "#000" /* black */
},
// Nur für Elemente mit der ID:"label"
"#label": {
        color: "#999" /* gray */
}
```

Die Gestaltungseigenschaften sind dieselben wie bei der traditionellen Entwicklungsmethode. Zur Definition von Styles gibt es drei verschiedene Arten: Elementeigene Styles, die auf alle Elemente eines Typs (beispielsweise Label) angewendet werden, einen Style zu einem bestimmten Element, welches über die ID bestimmt wird und Styles für eine Gruppe von Elementen (Class). Eine ID darf nur einmal in einem Dokument vorkommen. Um einen Style für eine Gruppe von Elementen zu entwickeln, benutzt man das Punkt-Präfix für den Style. Für ein einziges Element, welches über die ID bestimmt wird, nutzt man das Raute-Präfix. Bei Styles für einen Elementtyp verwendet man den Elementnamen. Wenn die Style-Typen gemischt werden, überschreibt der Style für das einzelne Element (ID) den Style für die Elementgruppe (Class) und dieser überschreibt wiederum den Style für einen Elementtyp. Im TSS-Code können eigene Variablen definiert werden, indem der Variablenname an „Alloy.CFG.“ angehangen wird („Alloy.CFG.Variablenname“). Weiterhin ist es möglich eine globale TSS-Datei anzulegen (styles/app.tss). Die Werte der globalen TSS-Datei werden von den lokalen Werten überschrieben, falls diese existieren. Um plattformabhängige Styles zu entwickeln, können die Attribute *platform* und *formFactor* eingesetzt werden:

[41]Vgl.: http://www.appcelerator.com/platform/alloy/ [23.07.2013]

```
"Label[platform=ios formFactor=handheld]": {
    backgroundColor: "#f00",
    text: 'iPhone'
},
```

Mit dem Alloy-Framework ist es möglich Themes für die Anwendung zu entwickeln. Dazu muss ein Verzeichnis namens „themes“ im app-Verzeichnis erstellt werden. Im Unterordner „assets“ können Dateien eingefügt werden, die bei gleicher Benennung und Nutzung dieses Themes die gleichnamige Datei im app-Verzeichnis „assets“ überschreiben. Um die Standart-Style-Datei durch die Theme-Style-Datei zu überschreiben, muss diese in den „styles“-Unterordner eingefügt werden. Auch hier kann eine globale Style-Datei „app.tss“ eingefügt werden. Weiterhin ist es möglich auch im „styles“-Verzeichnis betriebssystemspezifische Unterordner einzufügen, um plattformabhängige Styles zu nutzen. Um ein Theme einzusetzen, muss es in der Konfigurationsdatei „config.json“ konfiguriert werden:

```
    "global": {
        "theme":"mytheme"
    },
    "os:ios": {
        "theme":"green"
    },
    "os:android": {
        "theme":"blue"
    },
```

Die Oberfläche wird folgendermaßen entwickelt:

```
<Alloy>
	<TabGroup>
		<Tab title="Tab 1" icon="KS_nav_ui.png">
			<Window title="Tab 1">
				<Label>Label im ersten Tab</Label>
				<Button title="MeinButton" left="10%" top="10%"
onClick="doClick"></Button>
			</Window>
		</Tab>
		<Tab title="Tab 2" icon="KS_nav_views.png">
			<Window title="Tab 2">
				<Label>Label im zweiten Tab</Label>
			</Window>
		</Tab>
	</TabGroup>
</Alloy>
```

Hier sieht man die Anwendung von Tags wie auch in HTML üblich (, <body>, <head>). EventListener werden Elementen über die Eigenschaften zugeordnet. Im Beispiel sieht man einen

onClickListener für den Button mit dem Titel „MeinButton“. Über die Eigenschaft „onClick“ wird der Name der Funktion zugewiesen, die beim Betätigen des Buttons ausgeführt wird. Im Beispiel handelt es sich um die Funktion „doClick“. Die Funktion selber wird im Logikteil entwickelt:

```
function doClick(e) {
    alert($.label.text);
}
var dbalert = Titanium.UI.createAlertDialog({
    title: 'Keine Verbindung zur Datenbank',
    message: 'Es konnte keine Verbindung zur Datenbank hergestellt
     werden. Bitte erneut versuchen!',
    buttonNames: ['Ok'],
    cancel: 1
});
function onButtonPress(e){
     dbalert.show();
}
$.index.open();
```

Im Ausführungsteil können sowohl JavaScript als auch Titanium-Funktionen eingesetzt werden. Der Zugriff auf Elemente auf der Oberfläche im Hauptteil vom Gestaltungs- oder Ausführungsteil erfolgt über die Class oder die ID eines Elements und nicht wie bei der traditionellen Methode über die Referenz des Objekts. Um Elemente mit dem Alloy-Framework zu erstellen verwendet man den entsprechenden Tag des Elements: Um einen Button zu erzeugen nutzt man den Button-Tag: „<Button></Button>“. Die Eigenschaften des Buttons werden in der tss-Datei festgelegt:

```
"Button":{
    title: "Foobar",
    top: 0,
    width: Ti.UI.SIZE
}
```

Alle Elemente, die auch durch die herkömmliche Methode erstellt werden konnten, können durch den Einsatz des Elementnamens als Tag auch mit dem Alloy-Framework erstellt werden. In der herkömmlichen Entwicklungsweise nutzt man die Titanium-Methode *Titanium.UI.createButton()* zur Erstellung eines Buttons und legt die Eigenschaften direkt bei der Erstellung fest. So sind die Eigenschaften, zum Nachteil der Übersicht, in derselben Datei enthalten:

```
Titanium.UI.createButton({
    text: "Foobar",
    top: 0,
    width: Ti.UI.SIZE
});
```

Zusätzlich zu dieser übersichtlichen Art und Weise steht auch die Möglichkeit zur Verfügung, die Eigenschaften direkt in der Oberfläche festzulegen, indem die Attribute mit in den Tag des Elements eingebracht werden:

```
<Button title="test" left="10%" top="10%" onClick="doClick"></Button>
```

Weiterhin besteht die Möglichkeit, Vorlagen für Elemente zu erstellen, die häufiger verwendet werden. Dazu erstellt man eine XML-Datei (beispielsweise „foo.xml“) und erzeugt darin ein Element, welches man als Vorlage nutzen möchte:

```
<Alloy>
    <Button id='fooButton'>I am a foo button!</Button>
</Alloy>
```

Nun kann man Instanzen erzeugen, indem man den Require-Tag benutzt:

```
<Alloy>
    <Window layout="vertical">
        <Require id="button1" src="foo" type="view"/>
        <Require id="button2" src="foo" type="view"/>
        <Require id="button3" src="foo" type="view"/>
    </Window>
</Alloy>
```

Im Attribut „src“ muss man sich auf die Vorlage beziehen („foo“ für „foo.xml“). Durch die Vergabe einer eindeutigen „id“ kann später auf die einzelnen Elementen zugegriffen werden.

Da es manchmal notwendig ist, Elemente erst zur Laufzeit dynamisch zu erzeugen, kann die Elementerstellung auch durch den Logikteil vorgenommen werden:

```
var button = Alloy.createController('foo').getView();
button.bottom = 0;
button.title = Neuer Titel';
button.width = Ti.UI.SIZE;
button.addEventListener('click', doClick);
// Auf der Oberfläche einfügen:
$.index.add(button);
```

Um aus dem Logikteil auf den Controller und somit auf Elemente der gleichnamigen Oberfläche zuzugreifen, kann der Dollar-Operator ($) verwendet werden. Oberflächen müssen auch im Alloy-Framework mit der Methode .open() geöffnet werden. Des Weiteren werden Oberflächen

gegenüber der herkömmlichen Methode nicht alle in einer Datei verwaltet, sondern jede Oberfläche hat ihre eigene XML-Datei. Zusätzlich kann auch zu jeder Oberfläche eine eigene Logik entwickelt werden, die sich in der namensgleichen JS-Datei befindet.
Alle XML-Dateien müssen mit dem <Alloy>-Tag beginnen und mit dem </Alloy>-Tag beendet werden. In der index.xml besteht eine Besonderheit: Als Kind-Element des <Alloy>-Tags dürfen nur Windows, TabGroups und SplitWindows eingesetzt werden. Um Elemente zu verwenden, die nicht aus dem Titanium.UI-Paket stammen, muss das „ns“-Attribut verwendet werden um das jeweilige Titanium-Paket auszuwählen:

```
<View ns="Ti.Map" id="map"/>
```

Davon gibt es allerdings ein paar Ausnahmen wie Menu, VideoPlayer und MusicPlayer, bei denen das „ns“-Attribut nicht notwendig ist. Um plattformspezifische Elemente einzusetzen, muss die Plattform mit dem „platform“-Attribut festgelegt werden:

```
<NavigationGroup platform="ios,mobileweb"/>
```

Um abhängig vom Betriebssystem und vom Gerät (Tablet, Handheld) verschiedene Ausgaben zu erzeugen, kann das „platform“- und das „formFactor“-Attribut eingesetzt werden:

```
<Annotation title="Cupertino" platform='ios' formFactor='tablet'/>
<Annotation title="Redwood City" platform='ios' formFactor='handheld'/>
<Annotation title="Mountain View" platform='android'/>
<Annotation title="Palo Alto" platform='android,ios,mobileweb'/>
<Annotation title="San Francisco" platform='mobileweb'/>
```

Um abhängig vom Betriebssystem eine eigene Oberfläche einzusetzen, kann im „views“-Verzeichnis ein Unterverzeichnis mit dem Namen des Betriebssystems erzeugt werden.

5.3 Technische Möglichkeiten

Titanium stellt alle notwendigen Elemente zur Gestaltung der Oberfläche zur Verfügung. Des Weiteren können jedem Element Events zugewiesen werden. Auch stellt Titanium eine HTTP Client-Komponente (Titanium.Network.HTTPClient[42]) zur Verfügung, mit welcher auf Webressourcen wie die Serverschnittstelle zugegriffen werden kann. Durch die Titanium- Entwicklungsform (Html5, JavaScript, CSS) sind alle JavaScript-Funktionen zur Funktionalitätsentwicklung und alle CSS-Möglichkeiten zur Festlegung von Attributen und zur Positionierung möglich. Zusätzlich zu den Web-Elementen stellt das Titanium SDK viele eigene Elemente zur Verfügung. Bei der Kompilierung einer Anwendung werden die entsprechenden nativen Elemente des entsprechenden Betriebssystems eingesetzt. Auf diese Art und Weise besteht ein Zugriff auf alle Elemente, die auch bei der nativen Entwicklungsweise bereitstehen. Nicht alle Elemente können auf allen Plattformen eingesetzt werden. Wenn eine Anwendung für mehrere Plattformen entwickelt wird, sollte darauf geachtet werden, dass alle Elemente auf den gewählten Plattformen einsetzbar sind. Die Plattform-Kompatibilität kann in der Titanium SDK-Dokumentation (Abbildung 20 - Kompatibilität) nachgeschaut werden, wird aber auch im Titanium Studio beim Einsatz der Elemente angezeigt. Das Titanium Studio stellt für jedes Betriebssystem ein eigenes Verzeichnis bereit, in dem betriebssystemspezifische Elemente abgelegt werden können. Dort können beispielsweise Designvorlagen oder Grafiken hinterlegt werden (Abbildung 19 - Verzeichnisse für Betriebssysteme). Auch können dort

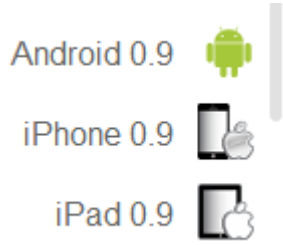

Abbildung 20 - Kompatibilität

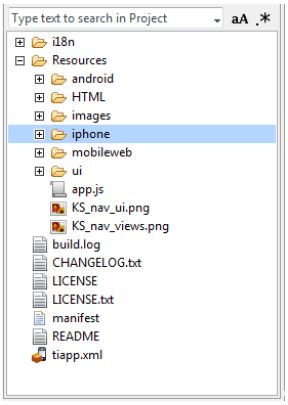

Abbildung 19 - Verzeichnisse für Betriebssysteme

```
client = Ti.Network.createHTTPClient(
      {
      onload : function(e)
            {
            // Antwort der Anfrage: this.responseText
            },
      onerror : function(e)
            {
            Ti.API.debug(e.error); //Debug-Ausgabe
            },
      timeout : 5000
      });
client.open("GET", geturl);
client.send();
```

Quellcode 1- HTTPClient

[42]Vgl.: http://docs.appcelerator.com/titanium/latest/#!/api/Titanium.Network.HTTPClient [25.07.2013]

JavaScripts abgelegt werden, die nur auf diesem Betriebssystem ausgeführt werden sollen (beispielsweise die Behandlung der fehlenden Zurück-Taste auf iOS).

5.3.1 Gerätezugriff: Dateisystem, Kamera, Sensoren, Karten, Bluetooth

Mit dem Titanium SDK ist der Zugriff auf unterschiedliche Sensoren, die Geräte-Kamera und das Dateisystem möglich. Um auf eine Datei aus dem Dateisystem zuzugreifen nutzt man die Titanium-Methode *Titanium.Filesystem.getFile()*. Da die verschiedenen Betriebssysteme die Dateien unterschiedlich verwalten, kann natürlich kein fester Dateipfad verwendet werden. Um einen universalen Dateipfad verwenden zu können stellt das Titanium SDK die Eigenschaft *Titanium.Filesystem.applicationDataDirectory* zur Verfügung. Um aus einer mit dem Titanium Studio erstellten Anwendung heraus die geräteeigene Kamera zu nutzen, wird die Methode *Titanium.Media.showCamera()* benutzt. Diese Methode kann auf die Events *success*, *cancel* und *error* reagieren. Bei *success* soll meist das aufgenommene Bild gespeichert werden. Dazu nutzt man die JavaScript-Methode *write()*, nachdem die zu beschreibende Datei mit der Titanium-Methode getFile() geöffnet wurde. Um auf die Ortungs-Sensoren der Geräte zuzugreifen, gibt es die Komponente *Titanium Geolocation*, die verschiedene Ortungs-Informationen sowie Kompass-Informationen bereitstellt. Diese Komponente kann auf iOS, Android, Mobile Web und Tizen eingesetzt werden. Über die Komponente *Titanium.Gestures* können Events wie der Orientierungs-Wechsel der Anwendung beim Drehen des Geräts und das Schütteln abgefangen werden. Um Bewegungen oder Beschleunigungen des Geräts erkennen und auswerten zu können, kann die Titanium-Komponente *Titanium.Accelerometer* eingesetzt werden. Um auf den betriebssystemeigenen Karten-Service zuzugreifen, steht die Titanium-Komponente *Titanium.Map* bereit. Diese wählt auf Android die Android Maps und auf iOS die Apple Maps aus. Die jeweilige Karte kann dann mit Positionierungs-Funktionen innerhalb der Anwendung verwendet werden. Das Titanium SDK enthält keine Bluetooth-API. Aus diesem Grund ist es notwendig fertige Module einzusetzen, wenn Bluetooth verwendet werden soll. Es existieren zwei Module für die Bluetooth-Nutzung: Das Bluetooth LE iOS Modul und das Bluetooth Android Modul. Diese Module sind nur für Android und iOS einsetzbar und kosten zusammen 700 US-Dollar.

5.3.2 Medienwiedergabe und –aufnahme:

Das Titanium SDK stellt zur Medienwiedergabe die Komponenten AudioPlayer[43], MusicPlayer[44]und VideoPlayer[45] bereit. Zur Audioaufnahme steht der AudioRecorder[46] zur Verfügung. Die Komponenten AudioPlayer und VideoPlayer sind kompatibel mit Android, iOS und weiteren Betriebssystemen. Der AudioRecorder und der MusicPlayer sind nur kompatibel mit dem iOS-Betriebssystem. Um Audioaufnahmen auf dem Android-Betriebssystem durchzuführen muss ein Fremdmodul verwendet werden[47]. Der MusicPlayer hat gegenüber dem AudioPlayer einen erweiterten Umfang. Mit ihm ist es möglich, eine Wiedergabeliste oder einzelne Mediendateien in einer Schleife wiederzugeben. Der AudioPlayer wurde nicht zur Wiedergabe von lokalen Audio-Dateien implementiert, sondern zum Streamen von Audio-Dateien aus dem Netzwerk. Allerdings kann der AudioPlayer zum Abspielen von lokalen Dateien verwendet werden, in dem der Pfad zur lokalen Datei im URL-Format angegeben wird („file:///mnt/sdcard/audio.mp3“). Hierbei handelt es sich um ein Workaround. Eine weitere Variante um Audiodateien auf dem Android-Betriebssystem auszugeben ist die Verwendung des VideoPlayers. Wenn reine Audio-Dateien wiedergegeben werden sollen, kann das Video-Fenster durch Anpassung der height-Eigenschaft versteckt werden. Im Gegensatz zum AudioPlayer ist hier die Wiedergabe von lokalen Dateien implementiert. Bei der Wiedergabe von Video-Dateien mit dem VideoPlayer muss für das Android-Betriebssystem der Mode bei der Instanzierung mit der *fullscreen*-Eigenschaft festgelegt werden. Der Mode kann anschließend nicht mehr geändert werden, für das Video wird eine eigene Activity im Vollbild-Modus erstellt. Auf dem iOS-Betriebssystem ist der Vollbild-Modus zur Laufzeit aktivierbar und deaktivierbar. Um einzelne Sound-Dateien wiederzugeben, die keine erweiterte Steuerung benötigen (Hinweis-Sound), stellt das Titanium SDK die Komponente Sound zur Verfügung. Um mit dieser Komponente eine Sound-Datei wiederzugeben, kann folgender simpler Code eingesetzt werden:

```
var player = Ti.Media.createSound({url:"sound.wav"});
player.play();
```

[43]Vgl.: http://docs.appcelerator.com/titanium/3.0/#!/api/Titanium.Media.AudioPlayer [27.07.2013]
[44]Vgl.: http://docs.appcelerator.com/titanium/3.0/#!/api/Titanium.Media.MusicPlayer [27.07.2013]
[45]Vgl.: http://docs.appcelerator.com/titanium/3.0/#!/api/Titanium.Media.VideoPlayer [27.07.2013]
[46]Vgl.: http://docs.appcelerator.com/titanium/3.0/#!/api/Titanium.Media.AudioRecorder [27.07.2013]
[47]Vgl.: http://www.codeboxed.com/2011/08/titanium-module-for-android-audio-recording/ [27.07.2013]

5.3.3 Animationen

Das Titanium SDK stellt eine Komponente zur Animation von Views bereit: Titanium.UI.Animation[48]. Es können alle View-Eigenschaften wie Größe, Position, Farbe und Deckkraft animiert werden. Die tatsächlichen Positions- und Größenwerte werden dabei nicht verändert und können nicht über die Get-Methoden abgefragt werden. Bei der Erstellung einer Animation sind ein Endwert und die Dauer der Animation anzugeben. Um eine Animation zu erstellen, wird die Methode Titanium.UI.createAnimation() verwendet. Weiterhin muss die Animation einer View zugewiesen werden, die animiert werden soll. Zusätzlich muss die Animation über einen EventListener gestartet werden. Das Beispiel erzeugt eine Animation, die die Hintergrundfarbe innerhalb von zwei Sekunden von rot über schwarz zu orange färbt.

```
var view = Titanium.UI.createView({
  backgroundColor:'red'
});
var animation = Titanium.UI.createAnimation();
animation.backgroundColor = 'black';
animation.duration = 1000;
var animationHandler = function() {
  animation.removeEventListener('complete',animationHandler);
  animation.backgroundColor = 'orange';
  view.animate(animation);
};
animation.addEventListener('complete',animationHandler);
view.animate(animation);
```

Abbildung 21 - Titanium Animation

5.3.4 OpenGL

Appcelerator stellt im Titanium SDK keine Komponente zur Nutzung von OpenGL bereit. Allerdings steht im Appcelerator Marketplace ein OpenGL-Modul für 29,99 US-Dollar zur Verfügung. Das Modul ist nur kompatibel mit iOS und kann nicht auf Android verwendet werden. Um das OpenGL-Modul nach dem Importieren zu nutzen, wird die Methode Ti.require('Ti.OpenGL') verwendet. Anschließend kann mit Ti.Opengl.createView() eine OpenGL-View erzeugt werden. Attribute der View können auf dieselbe Art und Weise wie bei anderen Titanium-Elementen zugewiesen werden. Es stehen die beiden Versionen OpenGL ES Version eins und zwei zur Verfügung. Nach der Erstellung der View können viele OpenGL-Methoden zur Erstellung von 2D- und 3D-Grafiken eingesetzt werden. Über *Ti.Opengl* und auch über damit erstellte Objekte können auf alle OpenGL-

[48] Vgl.: http://docs.appcelerator.com/titanium/3.0/#!/api/Titanium.UI.Animation [28.07.2013]

Standard-Methoden und -Elemente zugegriffen werden. Diese sind in der OpenGL-Dokumentation[49] zu finden.

5.3.5 Titanium Cloud Service

Beim Titanium Cloud Service handelt es sich um Mobile Backend as a Service (MBaaS) [50]. Durch die Cloud-Lösung ist es möglich sogenannte Push Notifikationen, Status Updates und einen Fotospeicher in der mobilen Anwendung zu nutzten. Des Weiteren ist die Anbindung an Sozial Media Dienste wie Facebook oder Twitter möglich. Bei Push Notifikationen handelt es sich um Meldungen, die auch bei aktueller Nichtnutzung der Anwendung auf dem Bildschirm eines Nutzers erscheinen wie beispielsweise eine SMS-Nachricht. Die Meldungen werden ausgegeben, wenn sich etwas innerhalb der Anwendung aktualisiert hat, wie beispielsweise beim Eingang einer neuen Nachricht.

Für die Nutzung der Titanium Cloud Services stehen drei Pakete bereit. Zum einen das Paket Titanium. Dieses Paket ist kostenfrei nutzbar, es enthält die Basis-Funktionen: Fünf Millionen Push-Notifikationen, fünf Millionen API-Aufrufe, zwanzig Gigabyte Speicherplatz und den Versand von 100.000 Emails pro Monat. Dieses Paket ist bei der Veröffentlichung einer Anwendung ausreichend. Wenn die Anwendung wirklich erfolgreich wird, kann das Paket „AppceleratorPlatform (Public Cloud)“ für 999 US-Dollar monatlich gebucht werden. Es enthält zehn Millionen Push-Notifikationen, zehn Millionen API-Aufrufe und 100 Gigabyte Speicherplatz sowie den Versand von 200.000 Emails pro Monat. Zusätzlich enthält dieses Paket weitere Funktionen wie das Performance Management, Funktionsprüfung und analytische Funktionen. Als letztes Paket, welches nur benötigt wird, wenn die Anwendung wirklich populär geworden ist, steht das Paket „Appcelerator Platform (Virtual private Cloud)“ zur Verfügung. Für dieses Paket fallen monatlich 2667 US-Dollar an. In diesem Paket können die Funktionen wie Speicherplatz, Versand von Emails, Push-Notifikationen und API-Zugriffe unbegrenzt genutzt werden. Weiterhin steht in diesem Paket der Support 24 Stunden täglich zur Verfügung und viele weitere Analysetools bereit.[51]

[49] Vgl.: http://www.opengl.org/sdk/docs/man2/ [29.07.2013]
[50] Vgl.: http://www.appcelerator.com/cloud/ [29.07.2013]
[51] Vgl.: http://www.appcelerator.com/plans-pricing/ [29.07.2013]

5.3.6 Appcelerator Marketplace

Die Entwickler des Titanium SDKs (Appcelerator Inc.) stellen mit dem Appcelerator Marketplace[52] eine Plattform zur Verfügung, auf der Entwickler sowohl eigene Module anbieten als auch erwerben können. Die Module können sowohl kostenfrei als auch kommerziell angeboten werden. Bei den Modulen handelt es sich um nützliche Erweiterungen für die entwickelten Anwendungen. Beispiele für die Module sind das Paypal Modul, das Barcode Scanner Modul und das Twitter Modul. Manche Module können plattformunabhängig eingesetzt werden während manche Module plattformabhängig sind (iOS 5 Twitter Modul, Bluetooth Android Modul). Das teuerste Modul im Store kostet 350 US-Dollar. Neben den Modulen werden auch Libraries angeboten. Hierbei handelt es sich um Sammlungen (Bilder, Musik,…). Eine Beispiel-Library ist die "App FX Sound Effects Library". Sie enthält 4000 Audio-Dateien. Die im Appcelerator Marketplace angebotenen Module unterteilen sich in die Kategorien *Erstellung, Verbindung, Testen, Verwaltung* und *Analyse*.

Nachdem ein Modul erworben wurde, kann dieses heruntergeladen werden. Anschließend wird das Modul in ein bestehendes Projekt importiert. Nach dem Import muss das neue Modul in die tiapp.xml der Anwendung aufgenommen werden. Anschließend kann das Modul mit `varModule = require('$MODULE_ID')` in der Anwendung benutzt werden.
Um selbst ein Modul zu erstellen, stellt das Titanium Studio den „New Mobile Module Project"-Dialog bereit, der nach Auswahl des Betriebssystems und Eingabe von einigen Informationen (Autor, Version, Copyright, Lizenz, Beschreibung) ein Modul-Projekt erstellt. Nach der Fertigstellung des Moduls kann dieses auf dem Appcelerator Marketplace veröffentlich werden, wenn folgende Bedingungen erfüllt werden[53]:

- Ein validierter Titanium Developer Account wird benötigt
- Die Werte in der Manifest-Datei müssen vollständig ausgefüllt werden
- In der Lizenz-Datei muss ein valider Lizenztext eingefügt werden
- In der index.md-Datei muss sich die Dokumentation zum Modul befinden
- Der Preis muss festgelegt werden (auch „kostenfrei" ist möglich)
- Eine Zahlungsmethode muss eingerichtet werden (bei kostenpflichtigen Anwendungen)

[52] Vgl.: https://marketplace.appcelerator.com/home [30.07.2013]
[53] Vgl.: https://wiki.appcelerator.org/display/tis/Creating+a+New+Titanium+Module [30.07.2013]

- Die „Terms of service agreement“ des Titanium Marketplace’s müssen akzeptiert werden

5.3.7 Appcelerator Analytics

Appcelerator Inc. stellt mit Appcelerator Analytics[54] eine Plattform zur Analyse der entwickelten Anwendung zur Verfügung. Die Analyse-Plattform zeigt sehr viele detaillierte Informationen, unter anderem die Anzahl der installierten Anwendungen, die Anzahl der aktiven Anwendungen und die durchschnittliche Nutzungsdauer der Anwendung an. Weiterhin wird angezeigt, auf welchen Geräten die Anwendung ausgeführt wird und welche Aktion die Benutzer am häufigsten durchführen um zu erkennen, welche Funktionen gefragt sind und welche nicht. Des Weiteren kann man die Anzahl der Installationen länderabhängig anzeigen lassen. Zusätzlich wird angezeigt, auf welchen Geräten und wie häufig die Anwendung abgestürzt ist. Dies hilft dem Entwickler Bugs zu finden und im folgenden Update zu entfernen.[55]

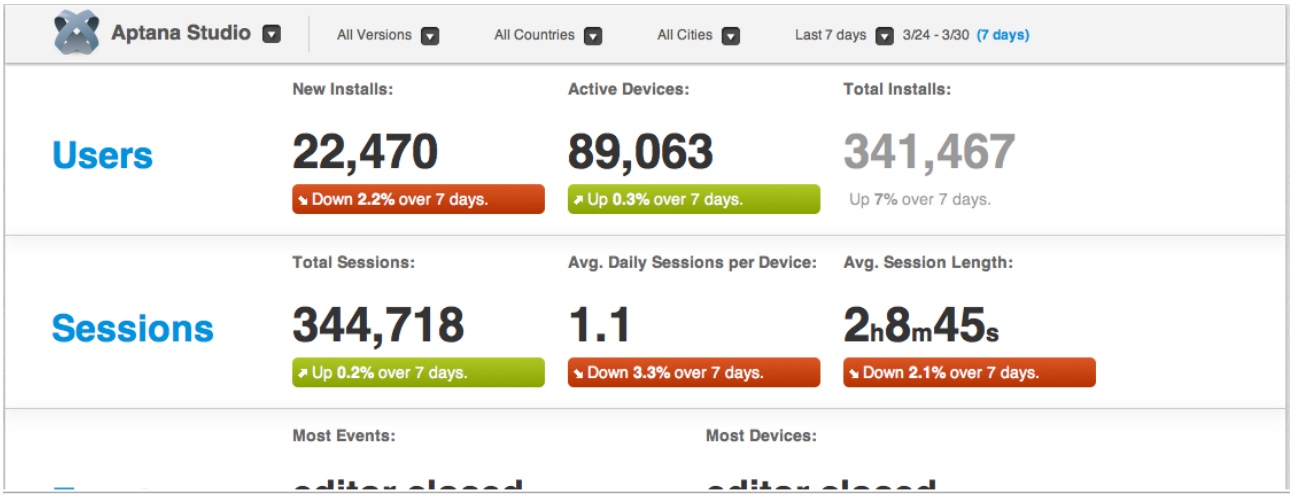

Abbildung 22 - AppceleratorAnalytics

[54] Vgl.: http://www.appcelerator.com/platform/appcelerator-analytics/ [30.07.2013]
[55] Vgl.: https://jira.appcelerator.org/secure/attachment/26755/Screen%20Shot%202012-03-31%20at%2012.22.51%20PM.png [30.07.2013]

5.4 Beispielanwendung „Mobile SQL-Trainer“ (Titanium-SDK-Edition)

In der Titanium-SDK-Edition des Mobile SQL-Trainers (Android) wurden die hauptsächlichen Funktionen der Anwendung entwickelt um diese mit den Funktionen der nativen Android-Version vergleichen zu können. Die vollständige Anwendung befindet sich in der app.js. Auch die Auslagerung einzelner Funktionen ist möglich. Für jede Ansicht wurde eine eigene Oberfläche (Window) entwickelt. Um zwischen den einzelnen Oberflächen zu wechseln wird die alte Oberfläche per *.close()*-Befehl geschlossen, und die neue Oberfläche mit dem *.open()*-Befehl geöffnet. Alle Anfragen an den Server (Aufgabe laden, Lösungsprüfung, Tabelle/Schema laden) werden mit der Ti.Network.HTTPClient-Komponente durchgeführt.

Abbildung 23 - Mobile SQL-Trainer (Titanium)

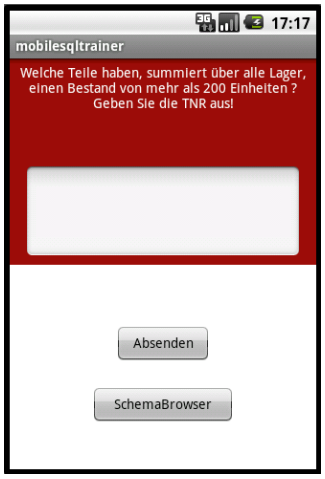

Abbildung 24- Aufgabe im Mobile SQL-Trainer (Titanium)

Nach dem Aufruf der Anwendung wird die Start-Oberfläche sichtbar (Abbildung 23 - Mobile SQL-Trainer (Titanium)).

Es ist erkennbar, dass das Titanium SDK eine effektive Positionierung sowie graphisch gestaltete Buttons und Auswahl-Picker zur Verfügung stellt. Hierbei handelt es sich um die nativen Android-Elemente, in die das Titanium-SDK die eingesetzten Elemente umgewandelt hat.

Abbildung 26 - Aufgabenergebnis (Titanium)

Nach der Auswahl der Einstellungen auf der ersten Oberfläche gelangt man über den „Test starten“ – Button auf die zweite Oberfläche (Abbildung 24- Aufgabe im Mobile SQL-Trainer (Titanium)). Auf der zweiten Oberfläche kann man erkennen, dass Elemente ohne CSS-Positionierung automatisch zentriert werden. Des Weiteren sieht man die von der Serverschnittstelle angefragte Aufgabenstellung. Während der

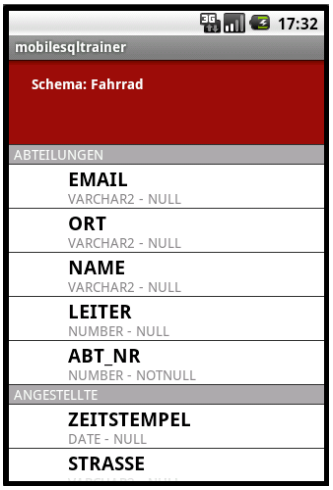

Abbildung 25 - Schema-Browser (Titanium)

Nutzung fällt auf, dass die Zugriffszeiten und die Performance dieser Anwendung sich nicht von der nativ entwickelten Anwendung unterscheiden. Die Kommunikation zwischen der Anwendung und der Serverschnittstelle funktioniert über HTTP-Anfragen von der Anwendung und HTTP-Antworten von der Serverschnittstelle. Die Antworten der Schnittstelle sind immer im JavaScript eigenen JSON-Format (JavaScript Object Notation) und können so sehr einfach von der Anwendung ausgewertet werden. Um eine Aufgabe lösen zu können ist es notwendig, den Schema-Browser zu nutzen. Im Schema-Browser werden dem Benutzer alle zum aktuell ausgewählten Schema gehörigen Tabellen mit deren Spalten und Spalteninformationen angezeigt (Abbildung 25 - Schema-Browser (Titanium)). Diese Oberfläche ist in 2 Views eingeteilt. In der oberen View befindet sich der Name des aktuellen Schemas. In der unteren View befinden sich die Tabellennamen mit den dazugehörigen Spalten. Diese werden in einer *Titanium.UI.ListView* verwaltet. Die ListView kann durch ein eigenes Template individuell angepasst werden.

Nachdem eine Aufgabe bearbeitet wurde, wird die Lösung überprüft und das Ergebnis der Überprüfung ausgegeben (Abbildung 26 - Aufgabenergebnis (Titanium)). Bei einer falschen Lösung wird der SQL-Fehler mit ausgegeben. Dieser wurde aus dem JSON-Objekt von der Serverschnittstelle extrahiert. Bei einer fehlerhaften Lösung kann der Benutzer auswählen, ob er einen erneuten Lösungsversuch unternehmen oder diese Aufgabe überspringen und den Test fortsetzen möchte.

5.5 Portierung auf iOS

Um eine mit dem Titanium SDK entwickelte Anwendung auf das Apple-Betriebssystem iOS zu portieren ist ein Mac-Gerät (MacBook, iMac) notwendig. Zusätzlich wird das iOS SDK benötigt.

Abbildung 29 - iOS-Simulator: Mobile SQL-Trainer

Wenn man eine Anwendung nicht nur im iOS-Simulator testen möchte, sondern auf einem reellen iOS-Gerät (iPhone, iPad) benötigt man zusätzlich einen Apple Developer Account. Für den Apple Developer Account fallen jährlich 99 US-Dollar an. Um das iOS SDK zu erhalten, muss die native iOS-Entwicklungssoftware Xcode aus dem App-Store installiert werden. Die mit dem Titanium SDK entwickelte Anwendung kann testweise direkt auf dem iPhone-Simulator ausgeführt werden. Natürlich sind die Elemente optisch nicht identisch mit den Android-Elementen, da hier native iOS-Elemente eingesetzt werden (Abbildung 29 - iOS-Simulator: Mobile SQL-Trainer). Die meisten Elemente (Buttons, Labels, Views) können ohne Änderung übernommen werden. Lediglich der Titanium.UI.Picker[56] muss größenmäßig angepasst werden, da sonst mehrere Einstellungsmöglichkeiten sichtbar sind (Abbildung 28 - Fehlerhafte Darstellung (iOS)). Leider funktioniert die Anpassung der Größe bei diesem Element nicht über das *height* oder *witdh* – CSS-Tag. Hier muss der Umweg über eine Titanium.UI.2DMatrix[57] gegangen werden, die dann über die Methode *scale*() skaliert wird. Über das Attribut *transform* des Pickers wird dann die *2DMatrix* zugewiesen und dadurch das ganze Element skaliert.

Abbildung 28 - Fehlerhafte Darstellung (iOS)

Abbildung 27 - AlertDialog

Des Weiteren darf die JavaScript-Funktion *alert* [58], die auf Android problemlos funktioniert, auf iOS nicht eingesetzt werden. Stattdessen kann sie durch das Titanium-Element Titanium.UI.AlertDialog[59] ersetzt werden. Dieser AlertDialog kann dann mit der Methode show()

[56] Vgl.: http://docs.appcelerator.com/titanium/3.0/#!/api/Titanium.UI.Picker [31.07.2013]
[57] Vgl.: http://docs.appcelerator.com/titanium/3.0/#!/api/Titanium.UI.2DMatrix [31.07.2013]
[58] Vgl.: http://de.selfhtml.org/javascript/objekte/window.htm#alert [31.07.2013]
[59] Vgl.: http://docs.appcelerator.com/titanium/3.0/#!/api/Titanium.UI.AlertDialog [31.07.2013]

geöffnet werden (Abbildung 27 - AlertDialog). Ein wesentlicher Unterschied zwischen der Entwicklung für Android und iOS ist die Art und Weise um zu einer vorherigen Oberfläche zurückzukehren. Auf älteren Android-Geräten existiert ein Hardware-Button um zurückzukehren, auf neueren Android-Geräten wird dafür ein Software-Button angezeigt.

Auf iOS existiert ein solcher Button nicht (Abbildung 31 – Schema-Browser(iOS)). Hier muss diese Funktion, wenn sie notwendig ist, vom Entwickler selbst implementiert werden. Wenn diese Funktion fehlt, gelangt der Anwender teilweise nur durch Beenden der Anwendung aus der Oberfläche heraus.

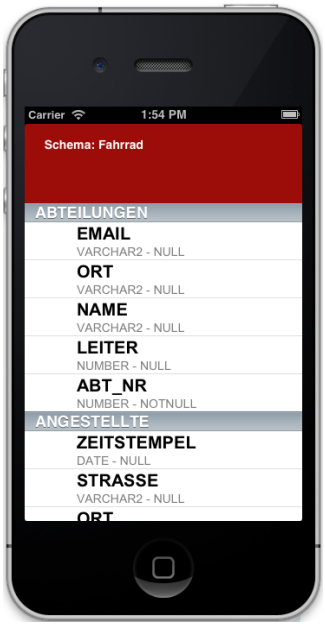

Abbildung 31 – Schema-Browser(iOS)

Die ListView kann unverändert übernommen werden (Abbildung 31 – Schema-Browser(iOS)) und wird vorteilhafter angezeigt als auf Android: Der aktuelle Tabellenname wird solange angezeigt, bis der nächste Tabellenname bis nach ganz oben gescrollt wird. Auf Android scrollt der Tabellenname direkt nach oben und es ist nicht mehr ersichtlich, zu welcher Tabelle die Spaltennamen gehören. In den weiteren Oberflächen der Anwendung sind keine Änderungen notwendig, die Elemente werden direkt korrekt dargestellt (Abbildung 32 - Endergebnis (iOS), Abbildung 30 - Aufgabenergebnis (iOS)). Die aufgetretenen Probleme konnten in kurzer Zeit behoben werden, so dass eine voll funktionsfähige iOS-Anwendung entstanden ist.

Abbildung 30 - Aufgabenergebnis (iOS)

Abbildung 32 - Endergebnis (iOS)

5.6 Portierung auf Mobile Web

Das Titanium SDK bietet die Möglichkeit an, die entwickelte Anwendung im Browser auszuführen. Dies ist vorteilhaft für nicht unterstützte mobile Betriebssysteme und kann zusätzlich in Browsern auf Desktop-PCs genutzt werden.

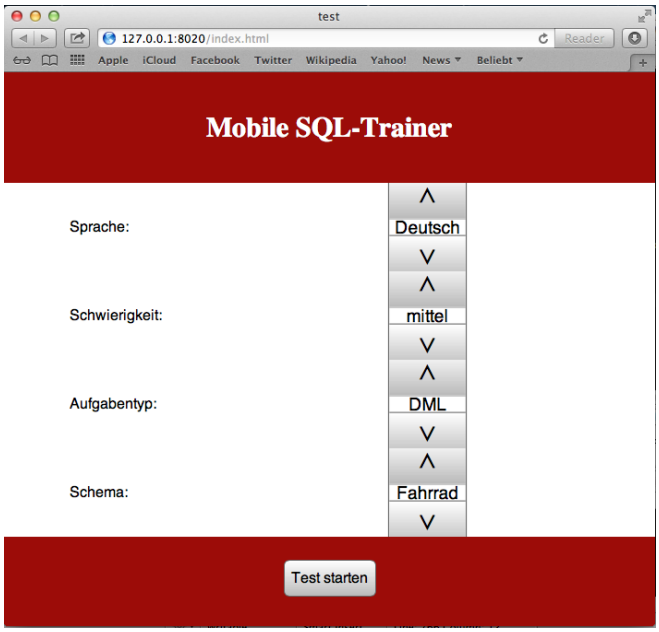

Abbildung 33 - Mobile Web : Mobile SQL-Trainer

Zum Testen einer entwickelten Anwendung stellt das Titanium SDK einen lokalen Webserver bereit, auf dem die Anwendung ausgeführt wird (Abbildung 33 - Mobile Web : Mobile SQL-Trainer). Die Picker mussten breitenmäßig angepasst werden. Auch wird deutlich, dass hier Html-Elemente eingesetzt werden. Da die Anwendung sich über den Titanium.Network.HttpClient[60] zur Serverschnittstelle verbinden soll, kann die Anwendung nicht weiter getestet werden, weil Zugriffe dieser Art durch die Same-Origin-Policy (SOP) untersagt sind und das Cross-Origin Resource Sharing nicht unterstützt wird (Abbildung 34 - Verbindung nicht möglich). Allerdings wird hier noch das selbst erstellte Hinweisfeld sichtbar, welches durch die Titanium.UI.AlertDialog-Komponente erstellt wurde.

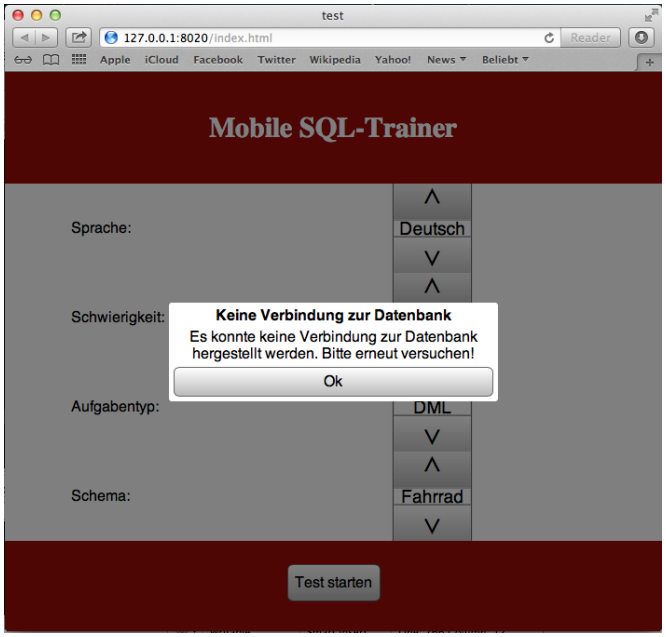

Abbildung 34 - Verbindung nicht möglich

[60] Vgl.: http://docs.appcelerator.com/titanium/latest/#!/api/Titanium.Network.HTTPClient [01.08.2013]

5.7 Vorschau: Portierung auf Tizen

Ein weiteres vom Titanium SDK unterstütztes mobiles Betriebssystem ist das Tizen-Betriebssystem[61]. Bei Tizen handelt es sich um ein von Intel, der Linux Foundation und Samsung entwickeltes Open-Source-Betriebssystem[62] für mobile Geräte. Tizen soll Smartphones, Tablets, Netbooks sowie Infotainmentsysteme für Autos und TV-Geräte unterstützen. Bisher ist noch kein Tizen-Smartphone erschienen, allerdings wurde die Entwicklung am 1.1.2013 von Samsung angekündigt. Auf Tizen sollen sowohl native Tizen-Apps als auch Bada-Apps und Android-Apps lauffähig sein[63].

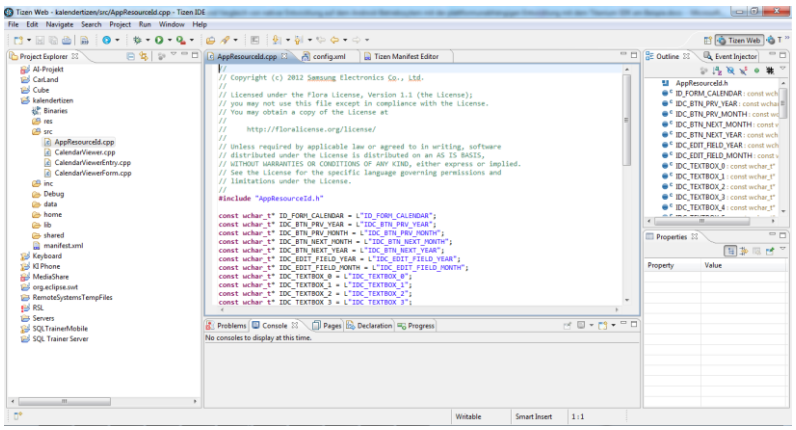

Abbildung 36 - Tizen IDE

Für die native Entwicklung stellt das Tizen SDK die Tizen IDE zur Verfügung, mit der außerdem Tizen Web-Projekte erstellt werden können (Abbildung 36 - Tizen IDE). Entwickelt wird in C/C++ bzw. JavaScript.

Weiterhin stellt das Titanium SDK einen Emulator-Manager (Abbildung 35 - Tizen Emulator Manager) mit einem Tizen-Emulator zur Verfügung. Es kann immer nur ein Emulator mit der aktuell installierten Tizen-Version gestartet werden. Ältere Versionen werden nur unterstützt, wenn ein x86-Disk-Image der Softwareversion zur Verfügung steht. Des Weiteren muss der Emulator vor einem Anwendungstest manuell gestartet werden. Um Anwendungen auf

Abbildung 35 - Tizen Emulator Manager

[61] Vgl.: https://www.tizen.org/ [01.08.2013]
[62] Vgl.: https://www.tizen.org/about [01.08.2013]
[63] Vgl.: http://www.chip.de/news/Tizen-Samsung-OS-laeuft-mit-Android-Apps_55869767.html [01.08.2013]

einem Tizen-Gerät (reell/virtuell) ausführen zu können, ist ein Autor-Zertifikat notwendig, welches mit dem im Tizen SDK enthaltenen Tool Certificate-Generator erzeugt werden kann.

Um eine Titanium-Anwendung auf Tizen zu portieren, muss das Tizen SDK installiert und im Titanium-Studio der Pfad zum Tizen SDK und zum Autor-Zertifikat konfiguriert werden. Nach Ausführung eines Tizen-Emulators kann die Titanium-Anwendung auf dem Tizen-Emulator installiert und gestartet werden. Wichtig hierbei ist, dass das Zertifikat korrekt eingetragen wurde und in der tiapp.xml-Datei die passende Titanium SDK-Version gewählt wird, da sonst die Anwendung nicht installiert und ausgeführt werden kann.

Um Web-Anwendungen mit graphischer Oberfläche ausführen zu können, werden momentan nur bestimmte Grafikkarten unterstützt. Aus diesem Grund ist ein umfangreicher Portierungstest nicht möglich, da der Emulator-Bildschirm nach dem Start der Anwendung nur ein schwarzes Bild ausgibt.

5.8 Vorschau: Portierung auf BlackBerry OS

Anwendungen, die mit dem Titanium SDK entwickelt wurden, können auch auf das BlackBerry OS portiert werden. Beim BlackBerry OS handelt es sich um das Betriebssystem des Unternehmens BlackBerry, welches für BlackBerry-Geräte entwickelt wurde. Mit dem BlackBerry SDK wird die QNX Monomentics IDE zur Entwicklung von nativen BlackBerry-Apps in C/C++ bereitgestellt. Zusätzlich zum SDK ist zur Entwicklung mit dem Titanium SDK der BlackBerry-Simulator notwendig. Der Simulator (Abbildung 37 - BlackBerry Simulator) wird, anders als bei den anderen SDKs, als virtuelles Image (VMware[64]) geliefert. Der Zugriff auf den Simulator erfolgt über die IP-Adresse der virtuellen Maschine. Wie auch bei iOS handelt es sich hier um keinen Emulator, sondern um einen Simulator, was unter Umständen zu Unterschieden auf einem

Abbildung 37 - BlackBerry Simulator

[64] Vgl.: http://www.vmware.com/de/ [02.08.2013]

reellen Gerät führen kann.

Zusätzlich zum BlackBerry SDK und dem Simulator ist eine Developer-Registrierung bei BlackBerry notwendig, um das BlackBerry Debug-Token zu erhalten.

Abbildung 38 - BlackBerry-Simulator: Titanium Logo

Die Entwicklung von BlackBerry-Anwendungen mit dem Titanium SDK ist erst mit der aktuellsten Version des Titanium SDKs (3.2.0) möglich, weshalb die Entwicklung noch nicht ausgereift ist. Nur durch Selbstversuche kann man zur geeigneten Kombination von Titanium SDK, BlackBerry SDK und BlackBerry-Simulator gelangen. Alle Komponenten müssen individuell konfiguriert werden. Im Beispielfall des Mobile SQL-Trainers startet die portierte Anwendung, das Titanium-Logo erscheint und die Anwendung wird wieder beendet (Abbildung 38 - BlackBerry-Simulator: Titanium Logo). Das Debugging ist nicht möglich, da an keiner Stelle eventuelle Fehlermeldungen ausgegeben werden. Aus diesem Grund ist ein ausgiebiger Test der Portierung auf das BlackBerry-Betriebssystem noch nicht möglich. Eventuell ändern sich diese Schwierigkeiten in zukünftigen Versionen des BlackBerry SDKs.

6 Vergleich der beiden Entwicklungsarten

6.1 Vorbereitung der Entwicklung

Um native Anwendungen (Android) in Eclipse zu entwickeln, muss Eclipse, ein JDK (Java Developer Kit) und das Android SDK installiert werden. Um eine plattformunabhängige Anwendung mit dem Titanium Studio entwickeln zu können ist das Titanium SDK und ebenfalls ein Java JDK sowie das jeweilige native SDK (beispielsweise das Android SDK) zu installieren. Der Unterschied hierbei liegt darin, dass das Titanium SDK bestimmte Versionen des Java JDKs, der installierten nativen SDKs und der installierten Emulatoren oder Simulatoren benötigt, da es sonst zu Problemen kommt. Eclipse ist in diesen Punkten anspruchsloser. Für die Erstinstallation des Titanium SDKs und der Entwicklungsvorbereitung für die einzelnen Plattformen wird viel Geduld benötigt, da nicht direkt klar hervorgeht, welche Versionen benötigt werden und Spezialanforderungen gestellt werden (beispielsweise JDK 1.6 32Bit-Version), die zu einem großen Teil nur durch Trial an Error herausgefunden werden können. Hilfeseiten zu Problemen (vor allem deutschsprachige) sind kaum vorhanden. Im Vergleich ist die Vorbereitung auf die Entwicklung mit Eclipse mit wesentlich geringerem Aufwand verbunden als die Vorbereitung auf die Entwicklung mit dem Titanium SDK.

6.2 Entwicklungsstil

Die Entwicklungsumgebung im Titanium Studio ist der Entwicklungsumgebung in Eclipse sehr ähnlich. Dies verwundert nicht, denn Titanium Studio basiert auf Eclipse. Leider wurde die LogCat aus Eclipse nicht direkt im Titanium Studio übernommen, allerdings werden Fehler, Informationen und Debug-Ausgaben bei der Nutzung des Emulators in der dazugehörigen Console ausgegeben. Wie detailliert diese sind ist unter anderem abhängig von der gewählten Plattform. Um eigene Debug-Ausgaben ausgeben zu lassen, nutzt man den Befehl *Titanium.API.info ("Beispielausgabe").*
Bei der nativen Entwicklung für Android mit Eclipse wird Java mit dem Android SDK verwendet. Um plattformunabhängig mit dem Titanium SDK zu entwickeln wird entweder JavaScript in Kombination mit Titanium SDK-Elementen oder dem Alloy-Framework benutzt, wobei die Entwicklung mit dem Alloy-Framework auf Grund einfacher Tag-Schreibweisen schneller voran geht. Java und JavaScript sind sich sehr ähnlich (Punkt-Operator, Wertzuweisung, Vergleich), der Einstieg in JavaScript gelingt mit Java-Kenntnissen sehr schnell.
Ein weiterer Unterschied zwischen den beiden Entwicklungsarten besteht in der graphischen

Gestaltung der Benutzeroberfläche. Bei der nativen Entwicklung mit Eclipse steht ein Drag&Drop-Tool zur vereinfachten Oberflächengestaltung zur Verfügung. Im Titanium Studio steht diese Möglichkeit leider nicht zur Verfügung, hier muss die Oberfläche codebasiert erstellt werden, was zeitaufwändiger ist. Im Vergleich ist die Oberflächen-Entwicklung mit dem Titanium SDK aufwändiger als mit Eclipse und die Debugging-Möglichkeiten in Eclipse sind umfangreicher. Dafür ist die gleichzeitige Entwicklung für mehrere Plattformen mit dem Titanium SDK möglich und der Zugriff auf Elemente der Oberfläche ist gegenüber der Android-Entwicklung mit Eclipse vereinfacht. Zusätzlich muss im Titanium Studio nicht für jede Oberfläche eine eigene Klasse angelegt werden, sondern alle Oberflächen können in einer Datei verwaltet werden.

6.3 Funktionalität

Mit Eclipse können zwar native Android-Anwendungen und HTML5-Anwendungen entwickelt werden, die Entwicklung für iOS, Tizen und Blackberry ist hiermit aber nicht möglich. Auch die Entwicklung eines Projekts, das anschließend auf mehreren Plattformen lauffähig ist, ist mit Eclipse nicht möglich.

Dagegen ist es mit dem Titanium SDK möglich Projekte zu entwickeln, die anschließend nach minimalen Anpassungen auf mehreren Plattformen (Android, iOS, (Blackberry), HTML5, (Tizen)) lauffähig sind. Die in Titanium eingesetzten Elemente werden bei der Kompilierung in die jeweiligen nativen Elemente übersetzt. Somit entsteht eine native Anwendung auf dem jeweiligen Betriebssystem und es kann auf geräteeigene Hardware-Funktionen zugegriffen werden. Die Entwicklung für Tizen und BlackBerry ist mit dem Titanium SDK noch nicht sehr ausgereift, und gelingt auch nach mehrtätigen Versuchen fast gar nicht. Konkrete Hinweise, warum eine Anwendung nicht ausgeführt werden kann, gibt es nicht.

Die Nutzung von Komponenten ist bei der nativen Entwicklungsweise mit Eclipse komplexer als bei der Entwicklung mit Titanium, was die Entwicklung auf der einen Seite erschwert, aber auf der anderen Seite auch mehr Möglichkeiten zur Individualisierung bietet. Eine Bluetooth-Verbindung kann bei der nativen Entwicklungsweise selbst erstellt werden, bei der Entwicklung mit Titanium ist diese nur durch den Einsatz eines kostenpflichtigen Moduls (Android und iOS: 700 US-Dollar) möglich. Andere Komponenten, wie das Dateisystem, die Gerätekamera, Gerätesensoren, und Karten können sowohl bei der nativen Entwicklungsweise als auch mit dem Titanium Studio ohne weiteres eingesetzt werden. Die Wiedergabe von Audio- und Videodateien sowie die Audioaufnahme sind in den gängigsten Formaten bei der nativen Entwicklungsmethode möglich.

Die Wiedergabe von Audio- und Videodateien ist mit dem Titanium Studio durch ein Workaround möglich, für die Aufnahme wird ein zusätzliches Modul benötigt. Beide Systeme unterstützen das Audio-Streaming. Die native Entwicklungsmethode stellt mehr Möglichkeiten zur Animation zur Verfügung, hiermit können sowohl Property- als auch View-Animationen erstellt werden. Mit dem Titanium SDK können nur View-Animationen erstellt werden, bei denen im Gegensatz zu Property-Animationen die Eigenschaften nicht tatsächlich verändert werden können. Die native Entwicklungsmethode stellt zusätzlich die Möglichkeit bereit, eine Bildfolge als Animation abzuspielen. Mit dem Android SDK können alle OpenGL-Funktionen in den Versionen eins, zwei und drei eingesetzt werden. Das Titanium SDK selbst unterstützt OpenGL nicht und mit dem OpenGL-Modul ist OpenGL nur auf iOS-Geräten einsetzbar. Weiterhin werden nur die OpenGL ES Versionen eins und zwei unterstützt. Sowohl bei der nativen als auch bei der Anwendungsentwicklung mit dem Titanium SDK stehen Libraries beziehungsweise Module zur Verfügung. Für die Entwicklung mit Java stellen Entwickler Libraries auf Ihren eigenen Internetseiten bereit, Module für Titanium werden im Appcelerator Marketplace angeboten. Sowohl Google als auch Appcelerator stellen mit den jeweiligen Cloud Services Möglichkeiten bereit um Push Notifikationen, Foto-Uploads und Status-Updates nutzen zu können. Beide sind bis zu einem gewissen Umfang kostenfrei nutzbar. Appcelerator stellt mit Analytics eine Möglichkeit bereit, veröffentlichte Anwendungen analysieren zu können. Eine ähnliche Möglichkeit steht mit der Integration von Google Analytics bereit.
Im Vergleich kann auf sowohl bei der nativen Entwicklung als auch bei der plattformunabhängigen Entwicklung fast gleichwertig auf Gerätefunktionen zugegriffen werden, da beide Entwicklungsarten im Endeffekt native Anwendungen produzieren.

7 Fazit

7.1 Zusammenfassung

Die mit Eclipse nativ entwickelte Android-Anwendung und die Titanium-Anwendung (Android und iOS) sind, zumindest mit unserem beispielhaften Funktionsumfang, fast gleichwertig. Die Entwicklung für BlackBerry und Tizen konnte auf Grund der noch nicht ausgereiften Entwicklungsart für diese Plattformen kaum getestet werden, allerdings wird der größere App-Markt mit iOS und Android unterstützt (Abbildung 39 - App-Store Vergleich)[65]. Die noch nicht ausgereifte Entwicklungsweise für das Tizen-Betriebssystem kann vernachlässigt werden, wenn bedacht wird, dass auf dem Tizen-Betriebssystem native Android-Anwendungen ausgeführt werden können. Außerdem wird Tizen noch auf keinem Smartphone eingesetzt (Stand 25.07.2013). Die Vorbereitung auf die Erstentwicklung ist durch die noch etwas weniger ausgereifte, aber dennoch sehr professionelle Entwicklungsumgebung beim Titanium SDK etwas zeitaufwändiger als bei der Eclipse-Entwicklungsumgebung. Natürlich sind die nativen und vom Betriebssystemhersteller herausgegebenen Entwicklungsformen der Vorreiter für das Titanium SDK und deshalb auch die sicherere Entwicklungsform, wenn es beispielsweise um Kompatibilität und Hardwarezugriff geht: Das Titanium SDK übersetzt den entwickelten JavaScript-Code in nativen Code der jeweiligen Plattformen mit den jeweiligen Plattform-Elementen. Die angepriesene Zeitersparnis von zwanzig Prozent bei der Entwicklung kann nicht bestätigt werden: Die Entwicklung der Benutzeroberfläche ist aufwändiger als bei der nativen Entwicklungsweise und die Dauer zur Funktionsentwicklung ist

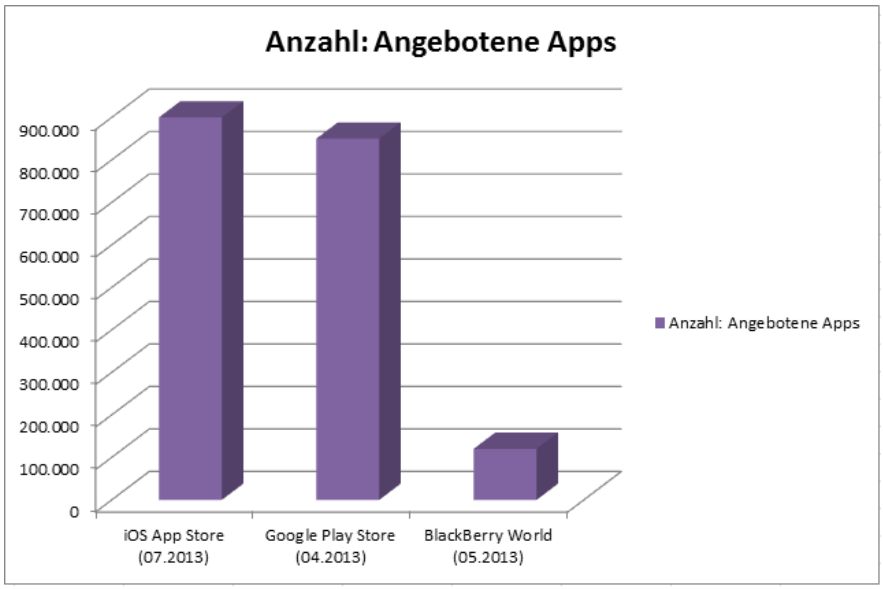

Abbildung 39 - App-Store Vergleich

[65] Vgl.: http://de.statista.com/statistik/daten/studie/208599/umfrage/anzahl-der-apps-in-den-top-app-stores/ [04.08.2013]

sehr ähnlich. Die Angabe zur Code-Wiederverwendbarkeit trifft durchaus zu: Die Beispielanwendung konnte durch wenige Änderungen auf das iOS-Betriebssystem portiert werden. Der Code konnte zu mehr als 90% wiederverwendet werden.

7.2 Empfehlung

Wenn es darum geht, eine Anwendung zu entwickeln, die mit den Standart-Elementen auskommt und auf Audioaufnahmen, OpenGL und Bluetooth verzichten kann, ist die Anwendungsentwicklung mit dem Titanium-SDK eine gute Alternative zur nativen Entwicklung. Die entwickelte Beispiel-Anwendung kann sowohl optisch als auch im Bereich der Performance mit der nativen Anwendung mithalten. Wenn dann noch eine plattformunabhängige Anwendung vor allem für die populärsten Betriebssysteme iOS und Android gewünscht wird oder eine der nativen Entwicklungssprachen (Java, Objective-C, C/C++) nicht beherrscht wird, ist das Titanium SDK voll und ganz zu empfehlen. Über die Probleme bei der Installation und die Einarbeitungsphase in die Kombination aus JavaScript und Titanium-Elementen kann bei den guten Ergebnissen hinweggesehen werden, zumal diese nur bei der ersten entwickelten Anwendung auftreten. Das erschwerte Debugging und die manuelle und dadurch aufwändigere Oberflächengestaltung sind Kritikpunkte, die allerdings in Kauf genommen werden können, wenn bedacht wird, dass durch einmalige Entwicklung Anwendungen für mehrere Plattformen entstehen. Die notwendigen Anpassungen um eine unter iOS lauffähige Anwendung zu erzeugen sind sehr gering. Allerdings sind zur Entwicklung einer nativen iOS-Anwendung, wie auch bei der nativen iOS-Entwicklung, Xcode, das iOS-SDK sowie ein Mac-Gerät und eine Apple Developer Lizenz (für 99US-Dollar) notwendig.

Dass die Entwicklung für Tizen und BlackBerry momentan noch fast unmöglich ist, ist kaum ein Nachteil: Tizen unterstützt native Android-Anwendungen und wird momentan noch nirgendwo eingesetzt und der BlackBerry-Markt ist sehr klein. Eventuell wird die Entwicklungsmöglichkeit für diese Plattformen aber in der Zukunft noch verbessert.

8 Literaturverzeichnis

Android Developers. (29. Juli 2013). Abgerufen am 29. Juli 2013 von http://developer.android.com/guide/topics/connectivity/bluetooth.html

Chip.de. (23. Juli 2013). Abgerufen am 23. Juli 2013 von http://www.chip.de/news/Tizen-Samsung-OS-laeuft-mit-Android-Apps_55869767.html

statista. (24. Juli 2013). Abgerufen am 24. Juli 2013 von http://de.statista.com/statistik/daten/studie/208599/umfrage/anzahl-der-apps-in-den-top-app-stores/

Tizen. (23. Juli 2013). Abgerufen am 2013. Juli 23 von https://www.tizen.org/about

Anonym. (8. Juli 2013). *Android Hilfe*. Abgerufen am 8. Juli 2013 von http://www.android-hilfe.de/android-app-entwicklung/21865-howto-android-programmierung-newbie-guide.html

Appcelerator Inc. (31. Juli 2013). *Alloy XML Markup*. Abgerufen am 31. Juli 2013 von http://docs.appcelerator.com/titanium/3.0/#!/guide/Alloy_XML_Markup

Appcelerator Inc. (05. Juli 2013). *Appcelerator - Titanium SDK*. Abgerufen am 05. Juli 2013 von Hauptseite: http://www.appcelerator.com/platform/titanium-sdk/

Appcelerator inc. (30. Juli 2013). *Appcelerator Alloy*. Abgerufen am 30. Juli 2013 von http://www.appcelerator.com/platform/alloy/

Appcelerator Inc. (31. Juli 2013). *Appcelerator Documentation*. Abgerufen am 31. Juli 2013 von http://docs.appcelerator.com/titanium/3.0/#!/guide/Views_without_Controllers

Appcelerator Inc. (9. Juli 2013). *Titanium Dokumentation*. Abgerufen am 9. 7 2013 von http://docs.appcelerator.com/titanium/latest/#!/api/Titanium.UI.Label

Google. (1. August 2013). *Aimation Resource*. Abgerufen am 1. August 2013 von http://developer.android.com/guide/topics/resources/animation-resource.html

Google. (29. Juli 2013). *Google Developers*. Abgerufen am 29. Juli 2013 von https://developers.google.com/maps/documentation/android/start#specify_settings_in_the_application_manifest

Google. (5. August 2013). *OpenGL - Android Developers*. Abgerufen am 5. August 2013 von http://developer.android.com/guide/topics/graphics/opengl.html

MAYFLOWER GmbH. (9. Juli 2013). *MAYFLOWER*. Abgerufen am 9. Juli 2013 von http://blog.mayflower.de/3643-Buch-Vorstellung-Titanium-Mobile-Multi-Platform-Apps-mit-JavaScript.html

Müller, D. (9. Juli 2013). *David Müller: Webarchitektur*. Abgerufen am 9. Juli 2013 von David Müller: Webarchitektur: http://www.d-mueller.de/blog/appentwicklung-webtechniken-mit-appcelerator-titanium/

Neumann, A. (11. Juli 2013). *heise Developer*. Abgerufen am 11. Juli 2013 von http://www.heise.de/developer/meldung/Titanium-Studio-Entwicklungsumgebung-fuer-plattformunabhaengige-Apps-1260783.html

SAMSUNG. (29. Juli 2013). *Samsung Developers*. Abgerufen am 29. Juli 2013 von http://developer.samsung.com/android/technical-docs/Sensors-in-Android

9 Eigenständigkeitserklärung

Ich versichere, die von mir vorgelegte Arbeit selbständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Gummersbach, den 08.08.2013

Matthias Spies

10 Anhang

Eclipse-Projekt Mobile SQL-Trainer (native Edition) mit mobilesqltrainer_eclipse.apk

Titanium-Projekt Mobile SQL-Trainer (Titanium-SDK-Edition) mit mobilesqltrainer_titanium.apk